VISUAL BASIC 6 FOR DUMMIES

Wallace Wang

APOGEO

Visual Basic 6 For Dummies®

Titolo originale:
Visual Basic 6 For Dummies®

Autore:
Wallace Wang

Viale Papiniano 38 – 20123 Milano (Italy)
Telefono: 02-461920 (5 linee r.a.) – Telefax: 02-4815382
Email **apogeo@apogeonline.com**
U.R.L. **www.apogeonline.com**

ISBN 88-7303-469-1

Traduzione di Silvia Bianchi e Pierpaolo Gardella
Impaginazione elettronica a cura di Edigramma – via Sacchini 20, Milano
Redazione di Mara Borgia

Indice generale

Introduzione

Ecco a voi la programmazione con Visual Basic 6. Se avete già provato a programmare e la cosa vi è sembrata troppo complessa, rilassatevi. Se siete in grado di scarabocchiare qualcosa su un pezzo di carta, allora siete anche capaci di scrivere un programma in Visual Basic 6 (non è uno scherzo).

A differenza di altri libri di programmazione che potreste avere preso e immediatamente lasciato perdere a causa della loro mole eccessiva, questo libro utilizza il metodo non convenzionale di parlare di programmazione in un italiano normale. Oltre a spiegazioni chiare e comprensibili a tutti in lingua italiana, questo libro fornisce anche informazioni generali, ma interessanti, sulla programmazione, un CD-ROM pieno zeppo di programmi d'esempio e di istruzioni per la scrittura passo dopo passo di programmi in Visual Basic 6.

Non date retta a quello che sentite dire in giro; la programmazione non deve essere necessariamente difficile, anzi, può essere persino divertente. Questo è un libro divertente e, dopotutto, il motivo che spinge la maggior parte delle persone ad acquistare un computer è divertirsi (ammettetelo, c'è davvero qualcuno che compra un computer solo per far quadrare un bilancio?).

Due parole sul libro

Considerate questo libro come una piacevole guida alla programmazione per Windows 95, 98 ed NT con Visual Basic 6. Sebbene Visual Basic non sia difficile da comprendere, ricordare tutte le minuzie necessarie a scrivere programmi in grado di fare qualcosa di interessante potrebbe essere faticoso e motivo di confusione. Tra gli argomenti trattati nel libro vi sono i seguenti.

- ✓ Salvare il programma.
- ✓ Progettare un'interfaccia utente.
- ✓ Creare dei menu a discesa.
- ✓ Eliminare gli errori di programmazione presenti in un programma.
- ✓ Stampare tutto.

Sebbene molti sostengano che per programmare un computer siano necessari anni di matematica superiore e di formazione tecnica, non spaventatevi. Se siete

in grado di spiegare a qualcuno come arrivare a casa vostra dall'ufficio, senza dubbio saprete anche dire al vostro computer che cosa deve fare. Lo scopo di questo libro non è quello di imbottirvi il cervello di dettagli tecnici e di spiegazioni, ma di illustrarvi tutti i passi indispensabili per scrivere un programma in Visual Basic e fornirvi le nozioni e la sicurezza necessari.

Come utilizzare il libro

In questo libro viene spiegato passo dopo passo come creare un vero programma Windows funzionante utilizzando Visual Basic. È sufficiente capire come si intende strutturare l'interfaccia utente e quindi scrivere il codice BASIC affinché il programma risponda a tutte le azioni compiute su quest'ultima, come per esempio fare clic sui pulsanti e premere determinati tasti.

Tutto il codice viene riportato con un carattere a spaziatura fissa, come quello che segue:

```
Printer.DrawWidth = Valore
```

A causa dell'impaginazione di questo libro, alcune righe lunghe di codice vanno a capo passando alla riga successiva. Il trattino di sottolineatura (underscore) al termine di una riga indica che il codice continua sulla riga sottostante.

```
Sub Form_MouseUp(Button As Integer, Shift As Integer, X _
          As Single, Y As Single)
```

Sul vostro computer, invece, le righe spezzate devono essere digitate come una singola riga di codice.

Per Visual Basic non ha importanza se scrivete tutto in caratteri maiuscoli, minuscoli o misti. Tuttavia, affinché ciò che compare sul vostro schermo corrisponda a quanto presentato sul libro, è opportuno utilizzare i caratteri maiuscoli e minuscoli impiegati negli esempi.

Potreste incappare anche in tabelle astruse come quella che segue.

Oggetto	*Proprietà*	*Impostazione*
Form	Name	frmCiao
	Caption	Ciao a tutti!
Option2	Name	optSorridi
	Caption	Sto bene

Quando si progetta un'interfaccia utente (come verrà spiegato con maggiori dettagli nel Capitolo 3), occorre definire le proprietà di ogni oggetto (come un pulsante di comando o una casella di controllo). La tabella presentata qui sopra comunica il nome dell'oggetto da modificare (come Form oppure Option2), le proprietà specifiche da cambiare (come Name o Caption) nonché il valore esatto da utilizzare in ogni particolare proprietà (come frmCiao oppure optSorridi).

Per verificare la vostra comprensione degli argomenti illustrati, nel libro sono inseriti dei semplici test sparsi qua e là. Questi test a scelta multipla non sono stati concepiti per farvi sentire degli incompetenti, ma per accrescere la vostra autostima. Tutte le possibilità tranne una sono talmente assurde che rispondere correttamente sarà molto facile. Non solo queste domande vi insegneranno qualcosa, ma leggendo le risposte sbagliate riuscirete anche a farvi qualche risata.

Per quanto vi riguarda

Se siete in grado di accendere e spegnere un computer, se sapete come utilizzare un mouse o una tastiera e inoltre desiderate scrivere i vostri programmi per divertimento, per soldi o per lavoro (non sempre il lavoro significa soldi o divertimento), allora siete pronti a scoprire come programmare in Visual Basic 6.

Oltre al vostro computer, dovete possedere una copia di Visual Basic 6 e desiderare di imparare a utilizzare il programma. Nel caso non siate sicuri delle vostre capacità di programmazione, ricordatevi che un insegnante delle scuole elementari frequentate da Albert Einstein (il famoso fisico che elaborò la teoria della relatività) era convinto che Einstein fosse ritardato.

Forse, in risposta a questa esperienza, Albert Einstein disse: "La fantasia è molto più importante della conoscenza". Alcuni affermano che Einstein disse anche: "Se il mio insegnante delle elementari è così bravo, dov'è il suo premio Nobel?". Ma quest'ultima affermazione non è mai stata confermata.

Quindi, se avete fantasia, un personal computer e Visual Basic 6, avete già tutto il necessario per cominciare a scrivere programmi in Visual Basic 6.

Com'è organizzato il libro

Questo libro è suddiviso in nove parti, ciascuna delle quali comprende alcuni capitoli a loro volta suddivisi in diversi paragrafi. Ogni volta che vi serve aiuto, aprite il libro e cominciate a leggere. Di seguito viene riportata una breve descrizione delle nove parti e dei relativi contenuti.

Parte I – Creare un programma in Visual Basic 6

La Parte I contiene una breve introduzione a tutte le principali caratteristiche di Visual Basic 6. Se l'idea di programmare un computer vi angoscia, qui potete superare l'ansia e aumentare la vostra autostima.

Parte II – Creare le interfacce utente

L'interfaccia utente determina l'aspetto del vostro programma. Questa è la parte divertente, dove potrete rendere il vostro programma brutto o bello quanto vorrete. Nella Parte II disegnerete cerchi e linee e scarabocchierete su tutto lo schermo. Se tentaste di fare la stessa cosa senza un computer, la gente potrebbe pensare che siete impazziti e regrediti all'infanzia.

Parte III – Creare i menu

Nella Parte III imparerete a creare splendidi menu a discesa degni dei più costosi programmi moderni. Se desiderate che il vostro programma faccia colpo sui vostri amici, in questa Parte vi insegneremo come fare.

Parte IV – I principi di base per la scrittura del codice

Nella Parte IV imparerete a scrivere dei genuini programmi BASIC che il computer capirà ed eseguirà. Benché possiate già sapere come dire al computer che cosa deve fare e dove deve andare utilizzando le parolacce, qui imparerete a farlo utilizzando il linguaggio BASIC.

Parte V - Prendere decisioni (qualcosa che smettete di fare quando vi sposate)

Nella Parte V scoprirete come dire al computer di decidersi a fare qualcosa invece di far decidere tutto a voi. Coloro che stanno crescendo i figli, troveranno questo aspetto particolarmente allettante.

Parte VI – Entrare in un ciclo

I cicli sono un altro sistema per dire al computer di fare qualcosa fino a quando la farà correttamente. Nella Parte VI imparerete diversi metodi per dire al vostro computer di ripetersi.

Parte VII – Scrivere sottoprogrammi (per non impazzire all'istante)

Molti si lasciano prendere dal panico all'idea di scrivere programmi di ampie dimensioni. Nella Parte VII imparerete a conservare la vostra salute mentale scrivendo tanti piccoli programmi che lavorano insieme per creare un grosso programma.

Parte VIII – File di database e stampa

La Parte VIII è intitolata "File di database e stampa", ma non lasciatevi intimidire dal titolo. I file di database sono solo dei metodi particolari utilizzati dai computer per organizzare le informazioni. Le persone archiviano le loro informazioni in cassetti, raccoglitori e armadietti; i computer utilizzano i database. Tutto qui. Inoltre imparerete a stampare i vostri preziosi programmi su carta, in modo che tutto il mondo possa vederli e ammirarli.

Parte IX – La parte dei 10

Nella Parte IX troverete alcune informazioni che potreste trovare utili e interessanti, tra cui suggerimenti su integrazioni a Visual Basic e su dove trovare ulteriori informazioni riguardanti la programmazione con Visual Basic.

Le icone utilizzate nel libro

Questa icona segnala dettagli tecnici utili (e talvolta interessanti) ma non indispensabili. Se volete, potete ignorarla.

Questa icona segnala informazioni utili che rendono la programmazione ancora meno complicata.

Non ignorate questi gentili promemoria: segnalano informazioni importanti.

Fate attenzione quando incontrate questa icona: vi segnala cose che potrebbero rovinarvi la giornata se non state attenti.

Questa icona segnala istruzioni e spiegazioni relative al codice.

Questa icona indica che è possibile cercare software e file di codice sul CD-ROM (che potrete caricare direttamente in Visual Basic invece di dover digitare tutto da zero).

Questa icona evidenzia le nuove caratteristiche di Visual Basic 6. Se sapete già utilizzare una versione precedente di Visual Basic, potete cercare queste icone per scoprire come utilizzare immediatamente le nuove caratteristiche di Visual Basic 6.

E adesso?

Nel caso vi stiate chiedendo cosa dovete fare adesso, la risposta ovvia è: girate la pagina e cominciate a leggere. Non preoccupatevi. La programmazione in Visual Basic 6 può essere divertente come giocare a Campo minato (con la sola differenza che in Visual Basic potete anche disegnare cerchi). Se qualcuno vi accusa di cincischiare invece di lavorare, potete sempre metterlo a tacere con tutte le vostre nuove conoscenze sulla programmazione in Visual Basic 6 e dirgli dove può andare partendo da lì.

Nello sforzo di divulgare Visual Basic tra le masse, Microsoft sta proponendone diverse versioni: la Learning Edition, la Standard Edition, la Professional Edition e la Enterprise Edition. Se utilizzate la Learning Edition (una versione limitata e a buon mercato studiata per consentire ai programmatori di procurarsi il programma con il minor esborso possibile) o la Standard Edition di Visual Basic, potreste non essere in grado di utilizzare tutti gli esempi descritti nei prossimi capitoli del libro.

Parte I

Creare un programma in Visual Basic 6

"UBALDO DIRIGE IL NOSTRO TEAM DI SVILUPPO SOFTWARE PER WINDOWS. AL MOMENTO STA LAVORANDO SU UN FOGLIO ELETTRONICO CHE È UN PO' UN MISTO TRA LOTUS 1-2-3 E NOSFERATU"

In questa parte...

Scrivere programmi non è poi così difficile. Se vi siete sempre interessati alla programmazione ma siete rimasti intimiditi dai libri difficili da leggere, dal software niente affatto utile o dalle oscure e involute "spiegazioni" degli "esperti", questo è il libro che fa per voi.

Invece di cercare di intimorirvi con teorie matematiche e con un background teorico sulla programmazione, questo libro vi permette, con l'aiuto di Visual Basic 5, di entrare immediatamente in argomento e di cominciare a scoprire da soli la programmazione sul vostro computer.

Quindi armatevi di un sacchetto di patatine, di una bevanda dietetica, sedetevi comodi e preparatevi a programmare il vostro computer in modo che alla fine faccia esattamente ciò che volete voi...

Capitolo 1

Come funziona Visual Basic

In questo capitolo

- Il ciclo di sviluppo dei programmi in Visual Basic
- Creazione di un'interfaccia utente in Visual Basic
- Che cosa fa il codice BASIC

Quando si scrive un programma, si vuole fare in modo che il computer faccia qualcosa di utile. Dato che spesso si spendono molti soldi per una di queste macchine, è importante che il computer non si limiti soltanto a consumare corrente elettrica e a occupare spazio su una scrivania.

Prima di tuffarvi nel mondo della programmazione, ricordate due cose. In primo luogo, tutti possono scrivere un programma. La programmazione è una capacità proprio come nuotare, navigare o taccheggiare. Se avete mai appreso un nuovo hobby o una nuova capacità, potrete anche imparare a scrivere un programma senza disporre di approfondite conoscenze di matematica o di una laurea di un'Università famosa e dieci anni di debiti per restituire il presalario.

In secondo luogo, il trucco della programmazione sta nel definire esattamente che cosa volete che il programma faccia. Arrivare a questo punto significa aver già vinto metà della battaglia (una battaglia che governi e società perdono costantemente). Per vincere anche l'altra metà occorre concedersi il tempo necessario per scrivere il programma e controllare che funzioni correttamente (questo è un altro scoglio contro cui i governi e le grosse società continuano a sbattere il naso).

È possibile scrivere programmi in modo da indurre il computer a fare qualsiasi cosa, tranne lanciare missili nucleari contro il vostro vicino (ovviamente, se scriveste un programma per i computer dell'Aviazione, potreste riuscire a fare anche questo).

Scrivere un programma Visual Basic

Non esiste un unico metodo per scrivere un programma. In teoria, esistono parecchi metodi diversi per scrivere un programma correttamente, proprio come esistono diversi metodi per viaggiare da New York a Los Angeles. Alcuni preferiscono

l'aereo, altri il treno, altri ancora l'automobile e i più avventurosi potrebbero andare a piedi, fare l'autostop o dirottare un aereo. Allo stesso modo esistono molti modi diversi per scrivere lo stesso programma e, a prescindere dal metodo utilizzato, il risultato può sempre essere lo stesso.

In qualità di programmatori, il vostro compito è quello di scrivere un programma che funzioni correttamente e sia semplice da utilizzare. Se il vostro programma non funziona, nessuno potrà utilizzarlo (sebbene potreste anche riuscire a venderne un po' di copie a persone ingenue). Se il vostro programma non è semplice da utilizzare, nessuno lo vorrà, neppure se funziona alla perfezione.

Verificare se il programma funziona è abbastanza semplice: se il vostro programma è concepito per stampare etichette e invece cancella tutti i file dal disco fisso, è ovvio che non funziona correttamente.

Stabilire invece se il vostro programma è semplice da utilizzare può essere più difficile: ciò che è semplice per voi potrebbe risultare quasi incomprensibile per qualcun altro.

Per aiutarvi a creare programmi che risultino semplici da usare per tutti, Visual Basic vi aiuta a realizzare senza fatica finestre, menu a discesa, finestre di dialogo e pulsanti di comando. Si tratta delle stesse caratteristiche presenti anche nei programmi Windows 95/98/NT. Visual Basic vi aiuta a scrivere programmi che assomigliano e si comportano come gli altri programmi presenti sul mercato.

Questo può aiutare gli altri a imparare i vostri programmi più rapidamente. Per esempio, la maggior parte delle persone è in grado di guidare indifferentemente una Toyota o una Ford perché il volante e i freni sono sempre allo stesso posto. Il fatto che alcuni accessori come i tergicristalli e il clacson si trovino in posizioni diverse non influisce sulla guida. Lo stesso vale per i programmi; i menu a discesa contengono i diversi comandi del programma, mentre il mouse viene utilizzato per evidenziare e selezionare comandi o oggetti. Quantunque ogni programma possa funzionare diversamente, la struttura generale rimane identica.

Il ciclo di sviluppo di Visual Basic

Prima di scrivere un programma in Visual Basic (o anche qualsiasi altro tipo di programma), allontanatevi dal computer e progettate il programma ricorrendo al vecchio metodo di carta e penna. Dopo che avrete capito cosa desiderate che il vostro programma faccia e che aspetto volete che abbia, potrete cominciare a scrivere il programma vero e proprio. Saltare questo passo fondamentale equivale a costruire una casa senza i disegni. Potete anche farlo, ma probabilmente impiegherete un tempo maggiore.

La stesura di un programma in Visual Basic è un processo in nove fasi. Le prime otto. Le prime otto fasi costituiscono ciò che i programmatori definiscono *ciclo di sviluppo*, mentre la nona rappresenta la *sicurezza del lavoro*.

Perché i programmi non funzionano?

Scrivere un programma che funzioni sempre correttamente al 100 percento è matematicamente impossibile. Innanzitutto, se scrivete un programma che funziona alla perfezione oggi, non avete alcuna garanzia che esso funzionerà alla perfezione sui modelli, processori e accessori dei computer futuri. Di conseguenza, è impossibile garantire che il programma funzionerà correttamente su tutti i tipi di computer, a meno che non abbiate testato in modo esaustivo tutte le possibili configurazioni.

In secondo luogo, occorre collaudare il programma utilizzando i prodotti più nuovi e sofisticati (compresi quelli non ancora inventati quando avete scritto il programma), ma dovete anche tenere presente il numero potenzialmente infinito di possibilità che il vostro programma deve affrontare durante l'utilizzo quotidiano.

Per esempio, il vostro programma deve comportarsi correttamente se l'utente preme qualunque tasto e poi fa clic con il mouse in qualunque punto dello schermo. Cosa succede se l'utente fa clic per errore mentre preme un tasto? Cosa succede se l'utente batte con forza sulla tastiera in preda alla frustrazione? Cosa succede se un altro programma interferisce con la memoria del computer influendo quindi sul vostro programma? Cosa succede se... (avete afferrato l'idea).

A meno che il programmatore sia in grado di pianificare un numero infinito di possibili problemi e situazioni che un programma potrebbe dover affrontare nel corso della sua esistenza, scrivere un programma che funzioni sempre al 100 percento, come già detto, è praticamente impossibile.

Quello che è più spaventoso è che questa è la realtà di ogni sistema operativo del mondo (come Windows 95, 98 ed NT), il che significa che verranno sempre scritti programmi da eseguire su un sistema operativo che non funziona correttamente al 100 percento. È come costruire una casa sulle sabbie mobili e poi continuare a chiedersi perché la casa continua a cadere.

Dato che nessuno dispone della quantità infinita di tempo necessaria per testare un numero infinito di possibili problemi, i programmi hanno sempre imperfezioni che impediscono loro di funzionare correttamente al 100 percento. Questo discorso vale per ogni programma che scrivete e per ogni programma che i programmatori milionari di Microsoft potrebbero scrivere. Per questo motivo, quando scrivete un programma dovete sempre verificarlo con molta attenzione, in modo da eliminare qualunque imperfezione potenzialmente pericolosa prima di dare il vostro programma a qualcun altro. Adesso sapete che la prossima volta che utilizzerete un programma che non funziona correttamente, sarete perfettamente giustificati nel biasimare i programmatori per non essere onnipotenti.

1. Decidere che cosa si desidera che il computer faccia.
2. Decidere che aspetto dovrà avere il programma sullo schermo (l'aspetto del programma viene definito *interfaccia utente*).
3. Disegnare l'interfaccia utente utilizzando elementi comuni quali finestre, menu e pulsanti di comando (gli elementi dell'interfaccia utente vengono definiti *oggetti* o *controlli*).

4. Definire il nome, il colore, le dimensioni e l'aspetto degli oggetti dell'interfaccia (le caratteristiche di un oggetto vengono definite *proprietà*).
5. Scrivere le istruzioni in BASIC in modo che ogni parte del programma faccia qualcosa (le istruzioni in BASIC vengono definite *comandi*).
6. Eseguire il programma per verificare se funziona.
7. Piangere disperatamente se il programma non funziona alla perfezione (obbligatorio).
8. Correggere gli errori presenti nel programma.
9. Ripetere infinite volte le fasi dalla 6 alla 8 finché vi stancate di cercare altri errori.

Sebbene non sia necessario ricordare a memoria queste nove fasi, è comunque necessario seguirle. Non esistono scorciatoie. Tentare di passare direttamente dalla fase 1 alla fase 4 equivale a tentare di avviare un'automobile utilizzando l'acceleratore senza infilare la chiave nell'accensione. Potete anche provarci, ma non andrete da nessuna parte.

Che ci crediate o meno, la fase 1 è la più difficile e anche la più importante. Dopo avere stabilito che cosa desiderate che il programma faccia, il processo di programmazione consiste semplicemente nel trovare i modi per far eseguire al programma i compiti desiderati. Tenacia e creatività sono di grande aiuto, proprio come le bevande contenenti caffeina e le numerose notti in bianco passate di fronte allo schermo del computer.

Creare una bella interfaccia utente

L'interfaccia utente è ciò che l'utente vede mentre il programma è in esecuzione. Ogni programma possiede un'interfaccia utente di qualche tipo. Alcuni programmi hanno finestre complesse e colorate, altri hanno un aspetto misero, come se il programmatore avesse temuto che prima o poi i fosfori dello schermo avrebbero cominciato a scarseggiare.

Un'interfaccia utente di Visual Basic comprende form e oggetti. Una *form* è semplicemente una finestra che appare sullo schermo. La maggior parte dei programmi Visual Basic possiede almeno una form, sebbene la maggior parte ne utilizzi parecchie.

Gli *oggetti* sono gli elementi che compaiono in una form (Figura 1.1), quali i pulsanti di comando, le barre di scorrimento, i pulsanti di opzione o le caselle di controllo. Un oggetto consente all'utente di impartire comandi al programma. Se volete, potete creare un programma con un'unica form e senza oggetti, ma il risultato non sarebbe né utile né interessante.

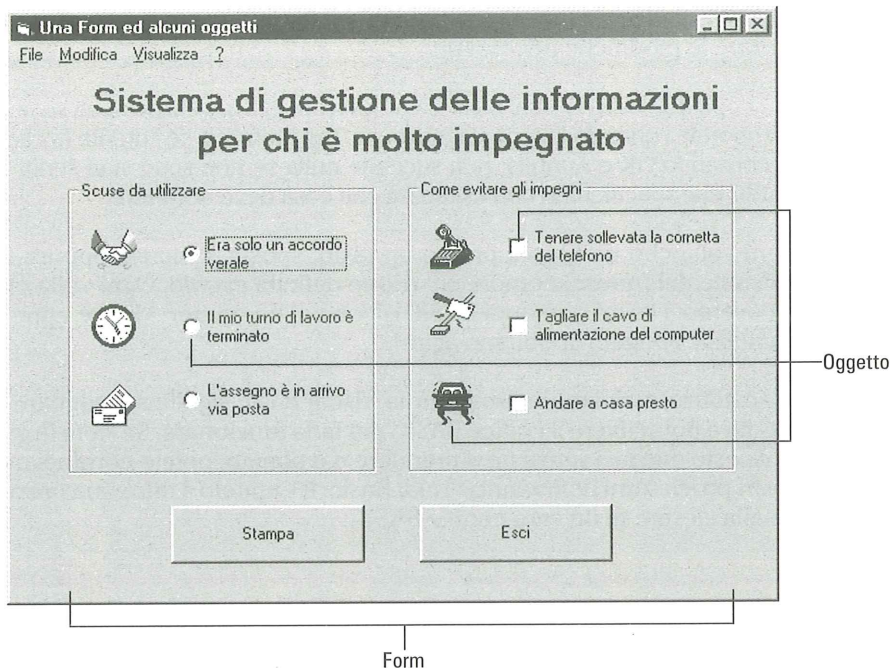



Figura 1.1 Una form e gli oggetti

Definire le proprietà che rendono bella l'interfaccia utente

Dopo avere creato una form e disegnato alcuni oggetti su di essa, la fase successiva consiste nel definire le proprietà di ciascuna form e oggetto. Le *proprietà* di un oggetto determinano il nome, il colore, le dimensioni, la posizione e l'aspetto dell'oggetto sullo schermo.

Oggetti diversi hanno proprietà diverse. Ogni volta che viene disegnato un nuovo oggetto, Visual Basic assegna alle proprietà dei valori di default che definiscono un oggetto generico che non può essere effettivamente utilizzato. Se volete personalizzare un oggetto, dovete definire una o più proprietà per ogni oggetto utilizzato dal programma. Il Capitolo 3 offre una rapida introduzione su come modificare le proprietà di un oggetto.

Scrivere un codice Basic

Quando siete soddisfatti dell'aspetto del vostro programma, la fase finale prevede la stesura dei comandi BASIC (noti anche come *codice*) per far funzionare effettivamente il programma. Non preoccupatevi. Se cambiate idea e desiderate modi-

ficare l'aspetto dell'interfaccia utente, potete sempre tornare sui vostri passi e apportare le modifiche necessarie.

Lo scopo del codice Visual Basic è comunicare agli oggetti presenti su una form come comportarsi quando l'utente fa qualcosa. Per esempio, se l'utente fa clic sul pulsante di comando OK o Annulla, non succede nulla se non sono stati scritti dei comandi BASIC che specifichino con esattezza che cosa deve accadere.

Ogni volta che un utente preme un tasto, sposta il mouse oppure fa clic con il pulsante del mouse, compie un'azione definita *evento*. Ogni volta che si verifica un evento, i vostri comandi BASIC dicono al computer: "Ehi, è appena successo qualcosa. Sbrigati, dobbiamo agire".

In sostanza, scrivere un programma Visual Basic significa disegnare l'interfaccia utente e poi scrivere il codice BASIC per farla funzionare. Se siete in grado di gestire queste due fasi senza farvi prendere dal panico, potete cominciare a scrivere i vostri programmi utilizzando Visual Basic. Il Capitolo 4 offre una breve introduzione alla stesura di un vero codice BASIC.

Perché i programmi non funzionano (Parte II)

La maggior parte dei programmi vengono scritti da programmatori professionisti che magari hanno studiato programmazione per anni. Questo significa forse che tutti i programmi scritti da loro funzionano? Ovviamente no. Questo è il mondo dei computer, dove nulla funziona come dovrebbe, ricordate?

Oltre al fatto che il livello di capacità tra programmatori professionisti può essere significativamente diverso, ai programmatori professionisti viene spesso chiesto di scrivere programmi per compiti che loro stessi non capiscono. Per esempio, i programmatori possono non sapere assolutamente nulla di contabilità e tuttavia essere chiamati a scrivere un programma che controlli il sistema di trasferimento elettronico dei fondi all'interno di una banca. Allo stesso modo, programmatori senza alcuna capacità o esperienza di volo possono essere chiamati a scrivere un programma che controlli l'atterraggio, il decollo e il volo di un jumbo jet 747. Altri programmatori ancora che non capiscono nulla di medicina possono trovarsi a scrivere un programma per controllare uno strumento medico che somministra dosi di radiazioni ai malati di cancro. Come è possibile lavorare in un campo di cui non si sa nulla e ciononostante ricevere compensi sbalorditivi? Facile, basta diventare programmatori.

Assumere un programmatore senza alcuna conoscenza del problema che deve risolvere è come assumere un traduttore perché traduca dal greco in francese quando in realtà questa persona non sa né leggere né scrivere in nessuna delle due lingue richieste. Detto questo, non è forse meraviglioso che gli aerei non cadono, le banche non perdono soldi e gli hotel non fanno confusione con le prenotazioni?

Capitolo 2

L'interfaccia utente di Visual Basic

In questo capitolo

- Caricare Visual Basic
- L'interfaccia utente di Visual Basic
- Aprire, chiudere e spostare le finestre
- Uscire da Visual Basic

Questo è il momento di fare un profondo respiro ed esaminare l'interfaccia utente del programma, prima di immergersi ulteriormente nelle complessità della programmazione in Visual Basic. Dopotutto, se non sapete utilizzare l'interfaccia utente di Visual Basic, non sarete in grado di fare molti passi in avanti nella stesura dei vostri programmi Visual Basic.

Per utilizzare Visual Basic, dovete conoscere le seguenti tre funzioni.

- ✓ Come caricare Visual Basic da Windows 95, 98 e NT.
- ✓ Come utilizzare Visual Basic per scrivere i vostri programmi.
- ✓ Come uscire da Visual Basic.

Caricare Visual Basic

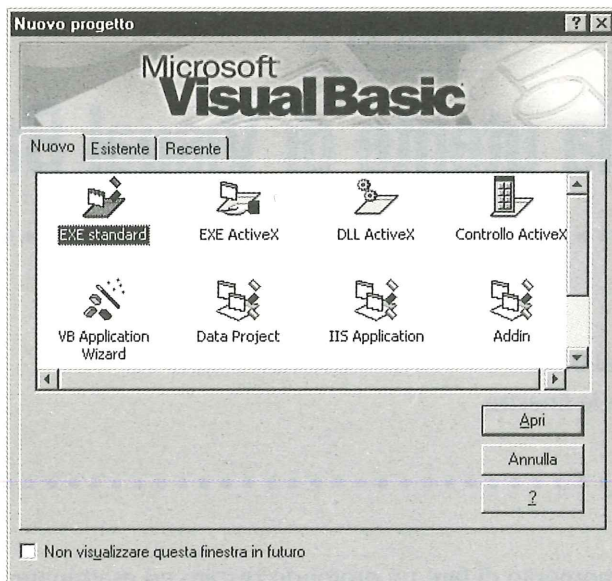
Per caricare Visual Basic, dovete seguire la semplice procedura di seguito riportata.

1. **Fate clic sul pulsante Avvio di Windows 95, 98, NT.**

   Appare un menu a discesa.

2. **Selezionate Programmi, fate clic sulla cartella Microsoft Visual Basic 6.0 (oppure sulla cartella Microsoft Visual Studio) e quindi fate clic su Microsoft Visual Basic 6.**

   Visual Basic visualizza la finestra di dialogo Nuovo progetto, illustrata nella Figura 2.1.

Figura 2.1 La finestra di dialogo Nuovo progetto che appare subito dopo avere caricato Visual Basic

Per caricare Visual Basic rapidamente, create una scorciatoia dal desktop sulle base di queste istruzioni: con il pulsante destro del mouse fate clic sul desktop di Windows, quindi fate clic su Nuovo, fate clic su Collegamento e quindi sul pulsante Sfoglia, ricercate il file VB6.EXE, fate clic sul pulsante Avanti, digitate **Visual Basic 6** e fate clic sul pulsante Fine.

Quando appare questa finestra di dialogo, avete diverse possibilità.

- Iniziare a scrivere un programma nuovo di zecca.
- Caricare un programma esistente per modificarlo.

Iniziare un nuovo programma

Quando la finestra di dialogo Nuovo progetto viene visualizzata, sono disponibili vari tipi di programmi da creare (la Learning e la Standard Edition di Visual Basic non visualizzano tutti questi tipi di programmi).

- **EXE standard.** Crea un programma stand-alone che potete copiare, distribuire oppure vendere. Esempi di programmi stand-alone sono Microsoft Word, Lotus 1-2-3 e Netscape Navigator; i programmi stand-alone hanno l'estensione di file .EXE.
- **DLL ActiveX.** Crea un file con l'estensione .DLL. Questi file non sono concepiti per essere utilizzati da soli, ma contengono sottoprogrammi progettati

per essere utilizzati come blocchi di costruzione quando si crea un programma stand-alone.

- **EXE ActiveX.** Crea un file con l'estensione .EXE. A differenza di un file .EXE stand-alone, questo file è progettato per lavorare come un server OLE, ovvero un programma concepito per condividere le informazioni con un altro programma.
- **Controllo ActiveX.** Crea un file con l'estensione .OCX. A differenza di un file DLL ACTIVEX o EXE ACTIVEX, questo file fornisce solitamente sia sottoprogrammi sia un'interfaccia utente che possono essere riutilizzati in altri programmi.
- **DDL documento ActiveX.** Crea un file con un'estensione .DLL. Tale file è concepito per aiutarvi a eseguire i programmi su un sito Web.
- **EXE documento ActiveX.** Crea un file con l'estensione EXE. Tale file può visualizzare una form Visual Basic all'interno di un browser Web su Internet.
- **AddIn.** Permette di creare un programma addin concepito per lavorare con l'interfaccia utente di Visual Basic.
- **Creazione guidata applicazioni VB.** L'utile guida di Visual Basic per aiutarvi a creare la struttura di un programma EXE stand-alone velocemente e facilmente.

- **Data Project.** Crea un programma per lo sviluppo di un database report che consente di visualizzare le informazioni contenute sul database in maniera chiara e organizzata.
- **DHTML Application.** Questo programma, una novità di Visual Basic 6, crea un documento DHTML (in linguaggio HTML dinamico) adatto ad essere collocato su un sito Web.
- **IIS Application.** Crea un file da utilizzare con Microsoft Internet Information Server - un'altra funzione nuova di Visual Basic 6.

Per selezionare una di queste opzioni dalla finestra di dialogo Nuovo progetto, fate clic sull'opzione desiderata e poi su OK.

Se selezionate File-Nuovo progetto (o premete CTRL+N), Visual Basic visualizza una finestra di dialogo Nuovo progetto che non contiene le schede Esistente e Recente.

La creazione dei file ActiveXIIS e Data Project presuppone delle conoscenze avanzate, quindi vi consigliamo di non prendete in considerazione questi file finché non avete capito come creare un semplice programma in Visual Basic (EXE Standard).

Caricare un programma esistente

Nella maggior parte dei casi caricherete un programma esistente per poterlo modificare. In questo caso, basta fare clic sulla scheda Esistente. Visual Basic visualizza le cartelle tra le quali potete scegliere il programma specifico da caricare.

Se volete caricare un programma che avete già caricato in precedenza, fate clic sulla scheda Recente. Visual Basic visualizza un elenco dei programmi che avete caricato di recente. A questo punto, fate clic sul programma Visual Basic che volete modificare e poi su Apri.

Se selezionate File-Apri progetto... (o premete CTRL+O), Visual Basic visualizza una finestra di dialogo Apri progetto che non contiene la scheda Nuovo.

Se fate clic sul menu File, Visual Basic visualizza un elenco dei programmi modificati di recente. Facendo clic su uno di questi nomi è possibile caricare il programma senza passare dalla finestra di dialogo Apri-progetto.

L'interfaccia utente di Visual Basic

Dopo avere deciso di iniziare un nuovo programma Visual Basic o di modificarne uno esistente, l'interfaccia utente di Visual Basic appare in tutto il suo splendore. Ovviamente, prima di poter disegnare l'interfaccia utente del programma e di scrivere del codice BASIC, dovete sapere come utilizzare l'interfaccia utente di Visual Basic, le cui otto parti principali sono illustrate nella Figura 2.2, anche se non tutte devono sempre essere visibili.

- **Menu a discesa.** Forniscono l'accesso a ogni comando Visual Basic disponibile, quantunque possano talvolta confondere le idee.
- **Barra degli strumenti.** Visualizza le icone che rappresentano i comandi Visual Basic utilizzati più di frequente; anche l'utilizzo di questa barra non è sempre immediatamente comprensibile.

- **Casella degli strumenti.** Visualizza i tipi di oggetti (come un pulsante di comando o la casella di controllo) che potete disegnare su una form. La casella degli strumenti potrebbe non elencare tutti i possibili oggetti che potete disegnare su una form. Per avere un elenco completo di tutti gli oggetti che potete visualizzare nella casella degli strumenti, premete CTRL+T.
- **Finestra progetto.** Elenca tutti i file che costituiscono un singolo programma Visual Basic.
- **Finestra (Proprietà).** Visualizza le proprietà della form o dell'oggetto selezionati in quel momento.
- **Finestra (Disposizione form).** Vi permette di modificare la posizione in cui le form appaiono sullo schermo.

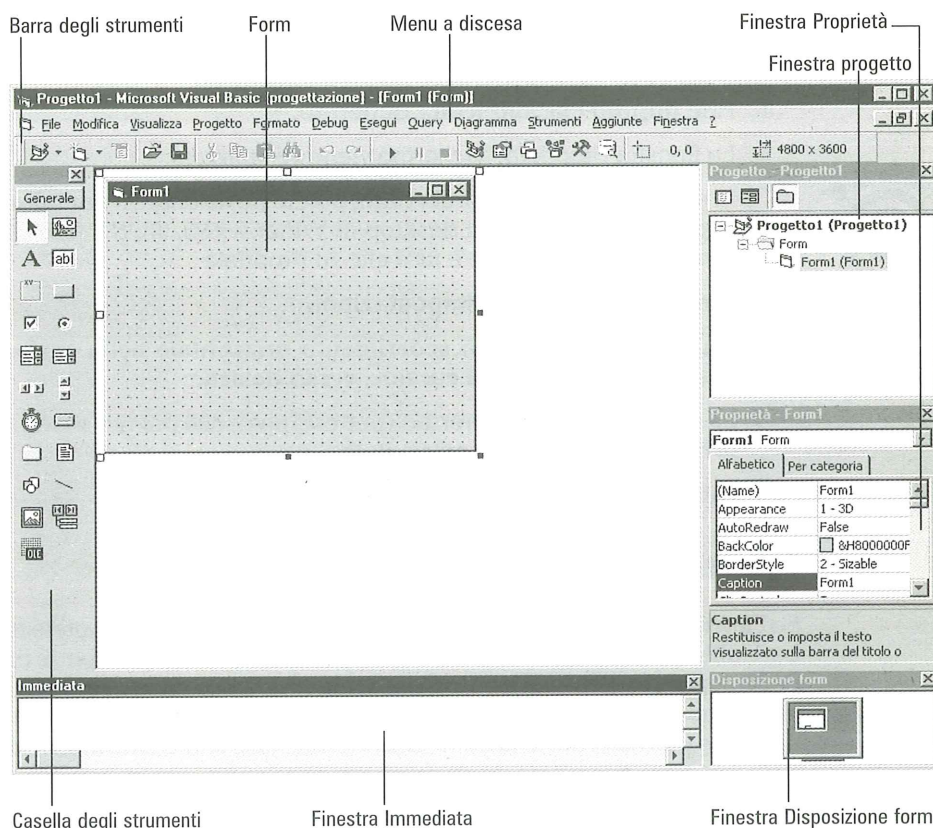



Figura 2.2 L'interfaccia utente di Visual Basic

- ✓ **Form.** Fornisce una finestra in cui potete disegnare gli oggetti per progettare l'interfaccia utente del programma.
- ✓ **Finestra Immediata.** Vi permette di eseguire il debugging del programma Visual Basic.

Progettando l'interfaccia utente utilizzate la casella degli strumenti per disegnare gli oggetti su una form. Dopo avere disegnato gli oggetti, la fase successiva consiste nel personalizzare l'aspetto di ogni oggetto utilizzando la finestra Proprietà. Infine, quando siete soddisfatti dell'aspetto dell'interfaccia utente, potete definire la posizione dell'interfaccia sullo schermo utilizzando la finestra Disposizione form.

Per visualizzare una parte maggiore dell'interfaccia utente di Visual Basic, modificate la risoluzione dello schermo seguendo la procedura di seguito riportata.

1. **Fate clic sul pulsante Avvio nella barra delle applicazioni di Windows 95, 98, NT che appare nell'angolo inferiore sinistro dello schermo.**

2. **Selezionate Impostazioni-Pannello di controllo.**

   La finestra Pannello di controllo viene visualizzata.

3. **Fate doppio clic sull'icona Schermo.**

   La finestra Proprietà - Schermo viene visualizzata.

   In alternativa a questa procedura, potete posizionare il puntatore del mouse in qualunque punto del desktop di Windows 95, 98, NT, fare clic con il pulsante destro del mouse e fare clic su Proprietà.

4. **Fate clic sulla scheda Impostazioni.**

5. **Fate clic sulla barra di scorrimento orizzontale nella sezione Area del desktop e trascinate il cursore verso destra.**

   La risoluzione dello schermo (per esempio, 800 per 600 pixel) appare sotto la barra di scorrimento orizzontale.

6. **Fate clic su OK.**

Chiudere e aprire le finestre

Se l'interfaccia utente di Visual Basic vi sembra troppo stipata, potete modificarla in base alle vostre esigenze senza alcun problema. Dato che gran parte degli elementi dell'interfaccia utente di Visual Basic è rappresentata da finestre, potete chiuderle per aumentare lo spazio disponibile sullo schermo. Poi, quando avete bisogno di utilizzare nuovamente le finestre dell'interfaccia, è sufficiente aprirle di nuovo.

Per chiudere una finestra nell'interfaccia utente di Visual Basic, fate clic sul pulsante di chiusura della finestra da chiudere. Per aprire una finestra e farla apparire nell'interfaccia utente di Visual Basic, selezionate Visualizza e poi fate clic sulla finestra che volete aprire, per esempio la finestra Codice o la Finestra Proprietà.

Ridimensionare e spostare le finestre

Invece di chiudere una finestra e farla scomparire completamente, potete anche ridimensionarla o disporla diversamente.

Per ridimensionare una finestra posizionate il cursore sul bordo della finestra finché il cursore assume la forma di una freccia a due punte. Poi tenete premuto il pulsante sinistro del mouse, trascinate il puntatore e rilasciate il pulsante sinistro quando la finestra ha la forma desiderata, come illustrato nella Figura 2.3.

Per spostare una finestra posizionate il cursore sulla barra del titolo della finestra, tenete premuto il pulsante sinistro del mouse, trascinate il mouse e poi rilasciate il pulsante sinistro quando la finestra si trova nella posizione desiderata.

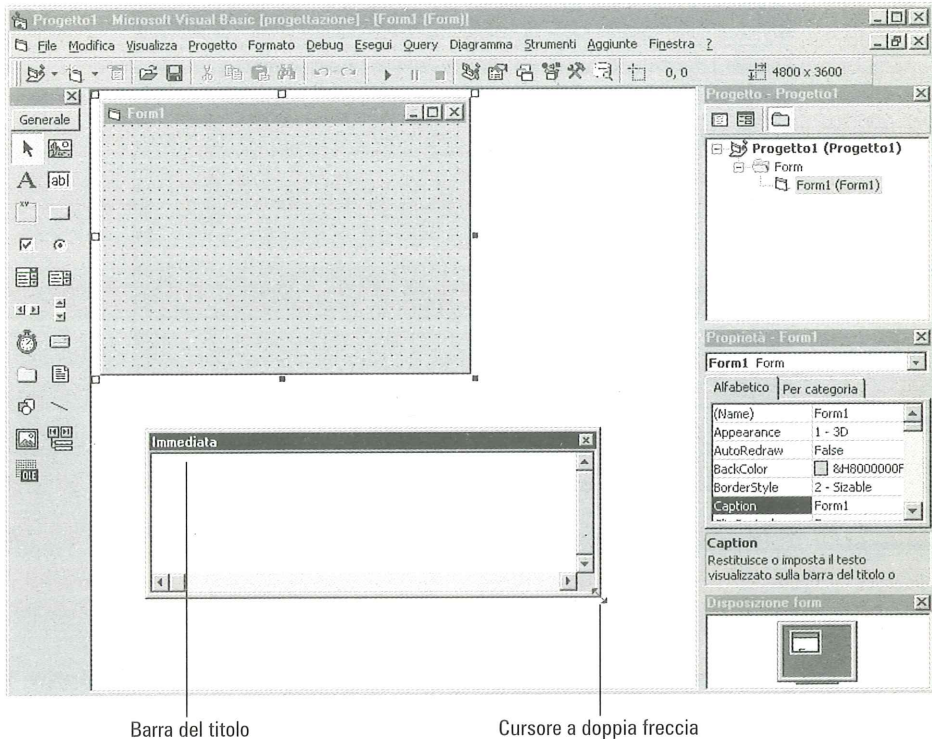



Figura 2.3 Ridimensionare una finestra

Ancorare le finestre

Le finestre che ingombrano lo schermo molto spesso si sovrappongono e sono d'intralcio. Per risolvere questo problema, Visual Basic offre una funzione ancoraggio.

Utilizzando questa caratteristica è possibile impilare ordinatamente le finestre su un lato dello schermo. A questo scopo, seguite la procedura di seguito riportata.

1. **Posizionate il cursore del mouse sulla barra del titolo della finestra che volete mettere in pila.**
2. **Fate doppio clic con il pulsante sinistro del mouse.**

   Visual Basic mette automaticamente in pila la finestra prescelta.

Nel caso preferiate che Visual Basic non utilizzi questa caratteristica potete disattivarla utilizzando la procedura di seguito riportata.

1. **Selezionate Strumenti-Opzioni.**

   La finestra di dialogo Opzioni appare, come illustrato nella Figura 2.4.

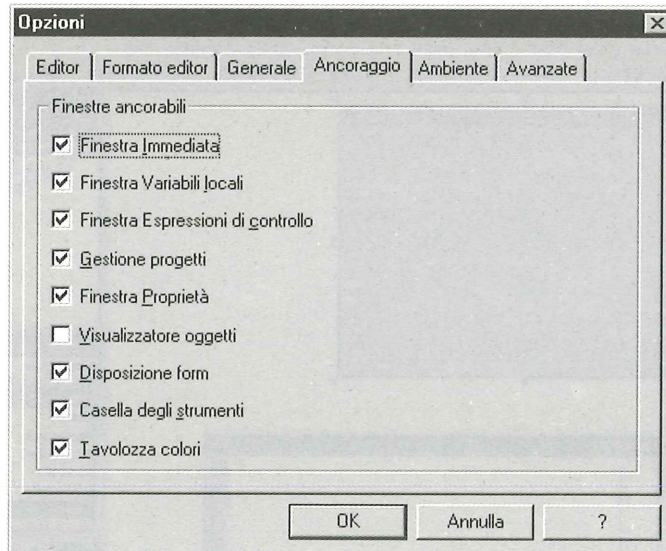



Figura 2.4 La finestra di dialogo Opzioni

2. **Fate clic sulla scheda Ancoraggio.**
3. **Fate clic nella casella di controllo per inserire o eliminare un segno di spunta dalla casella.**

   Un segno di spunta indica che la funzione ancoraggio è attivata per quella particolare finestra. La mancanza del segno di spunta indica che la funzione ancoraggio è disattivata.
4. **Fate clic su OK.**

Uscire da Visual Basic

A prescindere da quanto vi piaccia utilizzare Visual Basic, a un certo punto dovrete spegnere il computer e andare a dormire (o perlomeno perdere i sensi sulla tastiera per un'ora o due). Per uscire da Visual Basic utilizzate uno dei seguenti tre metodi.

- ✓ Selezionate File-Esci.
- ✓ Premete ALT+Q.
- ✓ Fate clic sul pulsante di chiusura della finestra dell'interfaccia utente di Visual Basic.

Se non avete salvato il programma Visual Basic visualizzato in quel momento, appare una finestra di dialogo che vi offre l'ultima possibilità di salvare il vostro lavoro prima che scompaia definitivamente. Fate clic su Sì per salvarlo o su No per perdere tutte le modifiche dall'ultima volta che avete salvato il file.

A meno che il vostro computer non abbia qualche problema, Visual Basic esce dolcemente riportandovi all'interfaccia utente di Windows 95, 98, NT.

Capitolo 3

Progettare la vostra prima interfaccia utente

In questo capitolo

- Gli elementi comuni di un'interfaccia utente
- Disegnare un'interfaccia utente
- Modificare le proprietà della vostra interfaccia utente

Come è già stato affermato nel Capitolo 1, occorre percorrere numerose fasi per creare un programma in Visual Basic dall'inizio alla fine. Mentre le prime fasi sono dedicate alla progettazione e alla creazione dell'interfaccia utente, questo capitolo si concentra sui passi fondamentali per la stesura di un programma in Visual Basic.

- ✓ Disegnare l'interfaccia utente.
- ✓ Definire le proprietà dell'interfaccia utente.
- ✓ Scrivere il codice BASIC.

Quindi, prima di poter scrivere il vostro primo programma in Visual Basic, dovrete imparare a creare un'interfaccia utente.

Gli elementi comuni di un'interfaccia utente

Nonostante le differenze, le interfacce utente grafiche tendono a condividere caratteristiche simili. Tutte le interfacce utente grafiche visualizzano testo o immagini in una *finestra*, che è un rettangolo che occupa l'intero schermo o una parte di esso. È possibile che due o più finestre appaiano contemporaneamente sullo schermo, sia sovrapposte sia affiancate. Per qualche strano motivo, in Visual Basic le finestre vengono chiamate *form*.

Appena creata, una form è completamente vuota. Per renderla utilizzabile, dovete disegnarvi degli *oggetti*; un oggetto può essere un pulsante di comando, una ca-

Una breve storia delle interfacce utente

Nei lontani anni '50, utilizzare un computer significava aprirlo e ridisporre alcuni cavi. Era necessario saper programmare un computer per utilizzarlo e saper collegare i cavi senza restare fulminati durante l'operazione.

Negli anni '60 per utilizzare un computer era necessario digitare i comandi per perforare le schede e poi inserire le pile di schede perforate nel computer. Dato che digitare perfettamente i comandi era un compito che la maggior parte delle persone considerava poco eccitante, usare un computer era un'attività lenta e noiosa.

Negli anni '70 collegarono un televisore a una tastiera e definirono questo insieme "terminale". Per la prima volta fu possibile digitare un comando direttamente sul computer e ottenere una risposta immediata. Questo fu il primo tentativo di creare un'interfaccia utente che le persone fossero in grado di utilizzare e capire effettivamente.

Ovviamente, queste prime interfacce utente prevedevano soltanto uno schermo vuoto e un puntino lampeggiante definito cursore. Perché il computer facesse qualcosa, era necessario digitare i comandi corretti; viceversa il computer si rifiutava di lavorare facendo sentire l'utente completamente stupido. Ancora una volta, usare il computer era un'attività lenta e noiosa.

In un tentativo disperato di rendere i computer più semplici da utilizzare, i programmatori inventarono ben presto qualcosa che battezzarono *interfaccia utente grafica* o *GUI* (Graphical User Interface). Fondamentalmente, una GUI visualizza i menu e le icone che l'utente può selezionare facendo clic sui comandi con il mouse.

Apple Computers creò la prima GUI commerciale quando introdusse il Macintosh, ma Microsoft creò velocemente una GUI propria (battezzata Microsoft Windows). Purtroppo, le GUI possono ancora rendere l'utilizzo del computer un'attività lenta e noiosa; quando questo succede prendetevela con il settore informatico. Questo non risolverà il problema, ma almeno vi farà sentire emotivamente meglio per uno o due minuti.

sella di testo, un'immagine o un pulsante di opzione. L'utente comunica con il vostro programma facendo clic, digitando o manipolando gli oggetti visualizzati su una form.

Da soli, gli oggetti non fanno assolutamente nulla, però sono carini. Per renderli funzionali, dovete scrivere un codice BASIC (di cui avrete maggiori informazioni nel Capitolo 4).

Disegnare oggetti con Visual Basic

Per disegnare gli oggetti su una form dovete utilizzare la casella degli strumenti che appare sul lato sinistro dello schermo (a meno che non l'abbiate spostata altrove). Questa casella contiene piccoli disegni rappresentanti i diversi oggetti che possono essere disegnati su una form, come illustrato nella Figura 3.1.

Figura 3.1 La casella degli strumenti di Visual Basic per disegnare gli oggetti su una form

Per disegnare un oggetto su una form, dovete sempre seguire la procedura di seguito riportata.

1. **Fate clic sull'oggetto che volete disegnare nella casella degli strumenti.**
2. **Posizionate il puntatore del mouse sulla form, nel punto in cui volete disegnare l'oggetto.**
3. **Tenete premuto il pulsante sinistro del mouse e trascinate il mouse per disegnare l'oggetto prescelto sulla form.**

Per disegnare più velocemente un oggetto sulla form fate doppio clic su un'icona nella casella degli strumenti; Visual Basic disegna automaticamente l'oggetto prescelto al centro della form.

I due metodi per creare un'interfaccia utente

Visual Basic fornisce due metodi per creare un'interfaccia utente.

- Utilizzare Creazione guidata applicazioni per creare un'interfaccia utente automaticamente.
- Creare un'interfaccia utente personalizzata partendo da zero.

Qual è il metodo migliore? Se dovete creare un programma Windows 95, 98, NT che offra i menu standard File, Modifica, Finestre e Guida, lasciate che sia Visual Basic a creare automaticamente l'interfaccia utente. Ma se state creando un semplice programma che non necessita dei menu a discesa, è più semplice creare l'interfaccia utente da soli.

A prescindere dal metodo prescelto, è sempre possibile aggiungere nuovi elementi all'interfaccia utente oppure cancellarla o modificarla in un momento successivo.

Disegnare l'interfaccia utente è un po' come fare scarabocchi. Nella Parte II troverete ulteriori informazioni sugli oggetti specifici e sul loro funzionamento. Per il momento è sufficiente che ricordiate due cose: tutti i programmi necessitano di un'interfaccia utente e Visual Basic utilizza le form e gli oggetti per aiutarvi a creare un'interfaccia utente in modo veloce e semplice.

Disegnare la vostra prima interfaccia utente da zero

Nel caso non abbiate voglia di percorrere tutte e quattordici le fasi, sul CD-ROM accluso troverete il file HELLO1.VBP memorizzato nella directory Chapter 3.

Per acquisire immediatamente dimestichezza con Visual Basic, ecco la procedura da utilizzare per creare una vera interfaccia utente partendo da zero.

1. **In Windows 95, 98, NT avviate Microsoft Visual Basic, se non lo avete già fatto. Se Visual Basic è già attivo, selezionate File-Nuovo progetto.**

   Visual Basic visualizza la finestra di dialogo Nuovo progetto nella quale dovete scegliere il tipo di programma che volete creare.

2. **Fate clic sull'icona EXE standard e poi su Apri.**

   Visual Basic visualizza una form vuota intitolata Form1 (fate riferimento alla Figura 3.1). Se volete che Visual Basic crei automaticamente un'interfaccia, fate clic sull'icona Creazione guidata applicazioni VB a questo punto della procedura, ma non fate clic sull'icona adesso. Alla fine di questo capitolo vedrete come utilizzare Creazione guidata applicazioni VB.

3. **Posizionate il cursore sull'angolo inferiore destro della form (direttamente sopra il rettangolino chiamato *handle,* che appare al centro del margine destro) in modo che il cursore si trasformi in una freccia che punta a sinistra e a destra, poi tenete premuto il pulsante sinistro del mouse e trascinate il mouse per ingrandire la form.**

4. **Selezionate Visualizza-Casella degli strumenti per visualizzare la casella degli strumenti sul lato sinistro dello schermo.**

   Saltate questa fase se la casella degli strumenti è già visibile.

5. **Fate clic sull'icona del pulsante Command sulla casella degli strumenti di Visual Basic, come illustrato nella Figura 3.2.**

6. **Spostate il mouse sulla form, quindi trascinatelo in modo da disegnare un pulsante di comando come quello illustrato nella Figura 3.2.**

7. **Fate clic sull'icona del pulsante Option sulla casella degli strumenti e disegnate un pulsante di opzione.**

   Ripetete quest'ultima operazione altre due volte per disegnare tre pulsanti di opzione, come riportato nella Figura 3.3.

Figura 3.2 L'icona del pulsante Command sulla casella degli strumenti di Visual Basic e un pulsante di comando disegnato su una form

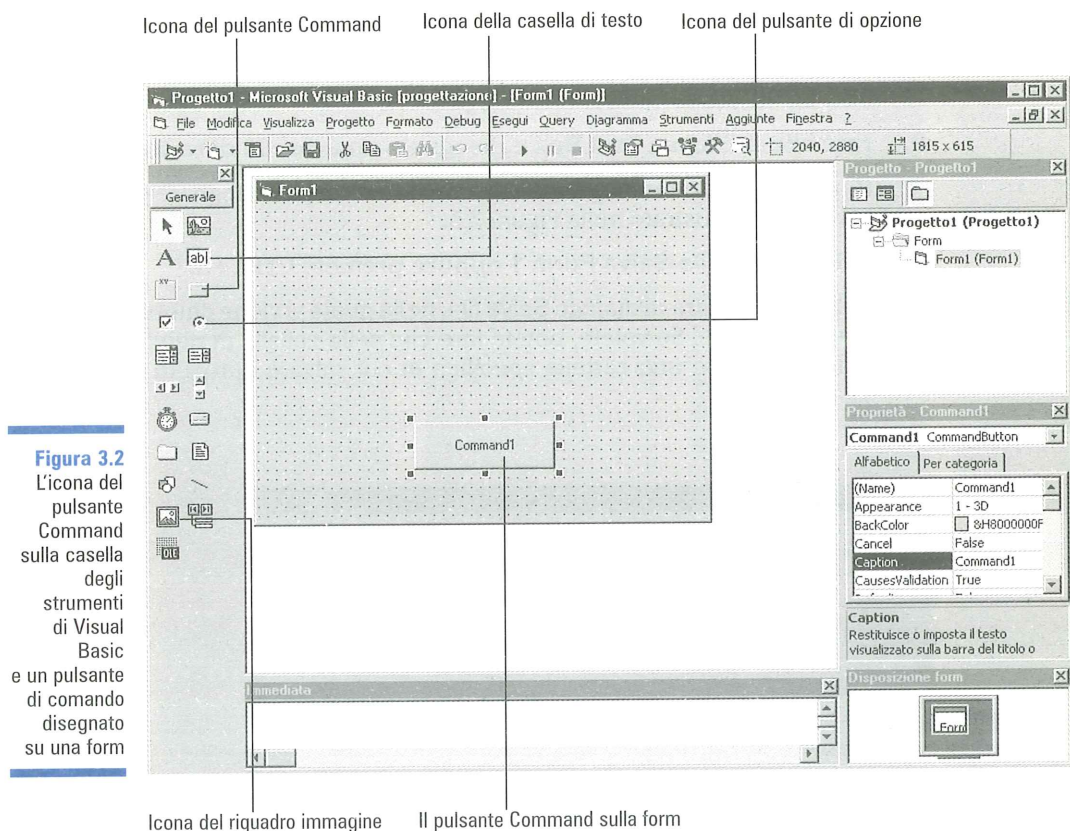
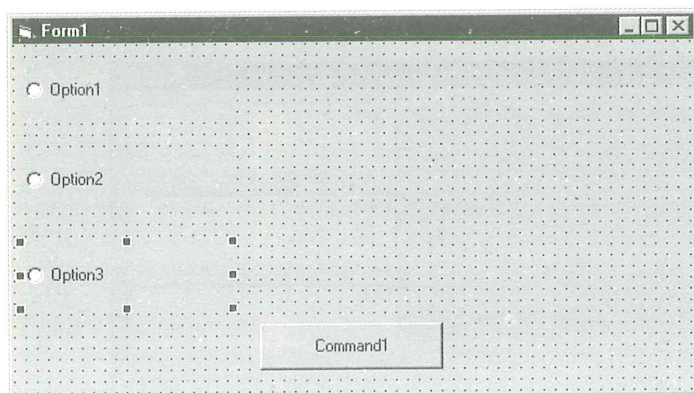



Figura 3.3 I tre pulsanti di opzione disegnati sulla form

8. **Fate clic sull'icona Image e disegnate un riquadro di immagine.**

   Ripetete quest'ultima operazione altre due volte per disegnare tre riquadri di immagine, come illustrato nella Figura 3.4.

9. **Fate clic sull'icona Text Box e disegnate una casella di testo.**

   Ripetete quest'ultima operazione altre due volte per disegnare tre caselle di testo, come illustrato nella Figura 3.5.

10. **Selezionate File-Salva Form 1 oppure premete CTRL+S.**

    Appare la finestra di dialogo Salva file, nella quale vi viene chiesto di indicare il nome del file. Se volete, potete fare clic nella casella di riepilogo Salva in per selezionare una cartella specifica in cui salvare i progetti e le form di Visual Basic.

Figura 3.4 I tre riquadri immagine disegnati sulla form

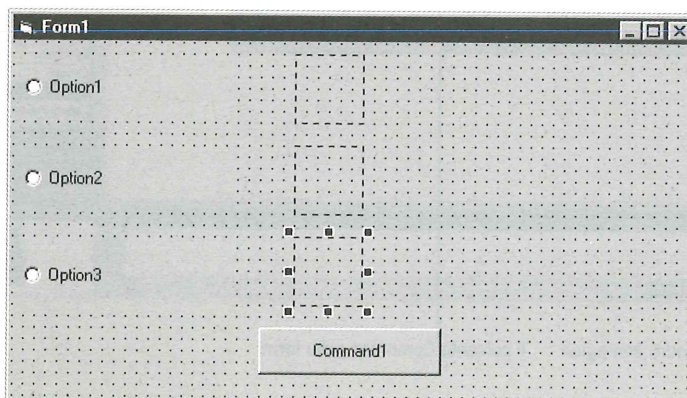
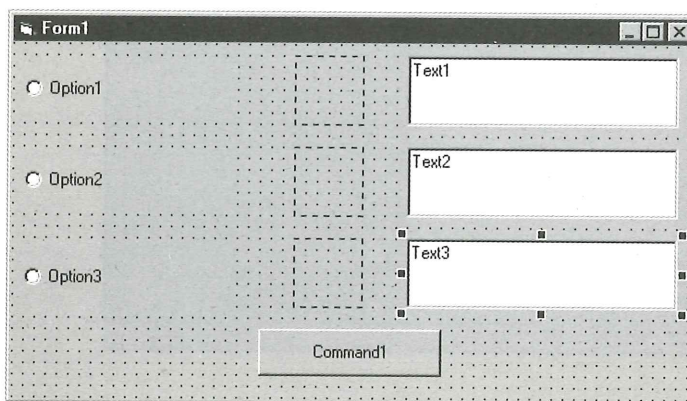



Figura 3.5 Le tre caselle di testo disegnate sulla form

11. **Digitate CIAO e fate clic su Salva.**

    In questo modo la vostra form viene salvata in un file CIAO.FRM.

12. **Selezionate File-Salva progetto.**

    Appare la finestra di dialogo Salva file con nome, nella quale vi viene chiesto di indicare il nome del progetto.

13. **Digitate CIAO e fate clic su Salva.**

    In questo modo il progetto Visual Basic viene salvato in un file denominato CIAO.VBP.

14. **Selezionate File-Esci (oppure premete ALT+Q) se volete uscire da Visual Basic.**

Congratulazioni! Avete appena creato un'interfaccia utente Visual Basic generica. Se non vi sembra proprio eccezionale è perché non è ancora stata personalizzata per il vostro programma.

Per personalizzare un'interfaccia utente Visual Basic, dovrete definire le proprietà di ogni oggetto presente sulle form.

Definire le proprietà della vostra interfaccia utente

Disegnare l'interfaccia utente significa compiere il primo passo nella scrittura di un programma Visual Basic. Il secondo passo consiste nel definire le proprietà di ciascun oggetto sull'interfaccia.

Ma è davvero necessario definire queste proprietà? Sì e no. Visual Basic imposta automaticamente dei valori di default per tutti gli oggetti presenti sull'interfaccia utente. Questi valori potrebbero però rendere poco attraente la vostra interfaccia utente. Quindi, se volete migliorarla, dovete definire le proprietà dei vostri oggetti.

Sebbene ogni oggetto abbia a disposizione da dieci a trenta proprietà che potete modificare, non dovrete modificarle tutte; di solito è sufficiente cambiarne una o due per oggetto.

Che cosa fanno le proprietà

Prima di modificare le proprietà di un qualunque oggetto, sarebbe meglio sapere su cosa tali proprietà influiscono. In linea di principio le *proprietà* definiscono le caratteristiche di un oggetto come il nome, le dimensioni, la forma e il colore sullo schermo.

Il nome di un oggetto viene specificato solo a vostro uso e consumo. Visual Basic fornisce automaticamente a tutti gli oggetti nomi noiosi come Text1 o Command3. Tuttavia, quando occorre riferirsi a oggetti specifici, tutto risulterà molto più semplice se utilizzerete nomi descrittivi per identificare i diversi oggetti utilizzati dal programma (nella Parte IV troverete ulteriori informazioni sulla scrittura del codice BASIC).

L'aspetto di un oggetto sullo schermo dipende dalle sue dimensioni, dalla posizione e dal colore. La definizione dell'aspetto di un oggetto permette di rendere più piacevole l'interfaccia utente.

Modificare le impostazioni delle proprietà

È possibile modificare le impostazioni di un oggetto in due momenti diversi.

- ✓ Durante la fase di progettazione.
- ✓ Durante la fase di esecuzione.

La *fase di progettazione* è il momento in cui disegnate l'interfaccia utente, prima dell'esecuzione del programma.

Di solito le proprietà dell'oggetto vengono modificate durante la fase di progettazione; la proprietà più importante da modificare in questa fase è il nome dell'oggetto.

La *fase di esecuzione* si verifica quando il programma utilizza il codice BASIC per modificare le proprietà di un oggetto mentre il programma si trova in esecuzione. Ovviamente, prima che il vostro programma possa modificare le proprietà di un oggetto, dovete scrivere il codice BASIC che sia in grado di dire esattamente al vostro programma quali proprietà modificare.

La modifica di una proprietà durante l'esecuzione vi consente di creare animazioni oppure di visualizzare dei messaggi sullo schermo, per esempio i messaggi di errore oppure i messaggi che avvertono l'utente che il computer sta per scoppiare.

Modificare le proprietà durante la fase di progettazione

Ogni oggetto dispone di diverse proprietà, come illustrato nella Figura 3.6. Per modificare una delle proprietà di un oggetto, assicuratevi che una form sia visualizzata in Visual Basic e seguite la procedura di seguito riportata.

1. **Fate clic sull'oggetto del quale desiderate modificare le proprietà.**
2. **Nella finestra Proprietà fate clic sulla proprietà che volete modificare.**

   Se la finestra Proprietà non è visualizzata, premete F4 per farla apparire.
3. **Digitate o selezionate un nuovo valore per la proprietà.**

Figura 3.6 La finestra Proprietà nella quale sono illustrate le molteplici proprietà della casella di testo Text3

Facile, non è vero? Quando dovrete modificare più proprietà di uno o più oggetti, questo libro vi presenterà una tabella simile alla seguente.

Oggetto	*Proprietà*	*Valore*
Form	(Name)	frmCiao
	Caption	Ciao a tutti!

Ecco che cosa significa questa tabella.

1. **Fate clic sull'oggetto Form.**
2. **Fate clic sulla proprietà (Name) nella finestra Proprietà.**
3. **Digitate frmCIAO per modificare il valore della proprietà (Name).**
4. **Fate clic sulla proprietà Caption nella finestra Proprietà.**
5. **Digitate Ciao a tutti! per modificare il valore della proprietà Caption.**

Definire le proprietà della vostra prima interfaccia utente

Nel caso non aveste voglia di seguire le prossime diciassette istruzioni, sul CD-Rom accluso troverete il file HELLO2.VBP memorizzato nella directory Chapter 3.

Per definire le proprietà della vostra interfaccia utente seguite la procedura di seguito riportata.

1. **In Windows 95, 98, NT avviate il programma Microsoft Visual Basic.**

   Visual Basic visualizza la finestra di dialogo Nuovo progetto. Se Visual Basic è già in funzione, selezionate File - Apri progetto.

2. **Fate clic sulla scheda Recente.**

Mettete alla prova le vostre nuove conoscenze

1. Quali sono i due elementi comuni di quasi tutte le interfacce utente?

a. La facilità d'utilizzo dell'interfaccia e un manuale di 500 pagine che spieghi quanto l'interfaccia sia facile da utilizzare.

b. Menu che nessuno è in grado di comprendere e comandi che nessuno sa utilizzare.

c. Icone prive di senso e testo che non spiega alcunché.

d. Form e oggetti.

2. Come si fa a modificare le impostazioni delle proprietà?

a. Non è possibile. Prima è necessario che le impostazioni desiderino cambiare.

b. Con continue minacce e intimidazioni.

c. Utilizzando la finestra Proprietà mentre si progetta l'interfaccia utente (*fase di progettazione*) oppure scrivendo un codice BASIC per modificare le proprietà mentre il programma è in esecuzione (*fase di esecuzione*).

d. Applicando una calamita di fianco al monitor finché le immagini si deformano e distruggono lo schermo del computer.

3. **Fate clic sul file CIAO e poi su Apri.**

   Visual Basic carica la form CIAO.FRM e la visualizza sullo schermo.

4. **Fate clic su Option1 per evidenziare il pulsante di opzione; in questo modo dei piccoli rettangoli appaiono attorno ai bordi del pulsante di opzione.**

5. **Fate clic nella finestra Proprietà, fate clic sulla proprietà (Name) e digitate optFelice. Fate clic sulla proprietà Caption e digitate Sono felice!**

6. **Fate clic sul riquadro di immagine Image1 (quello in cima alle form).**

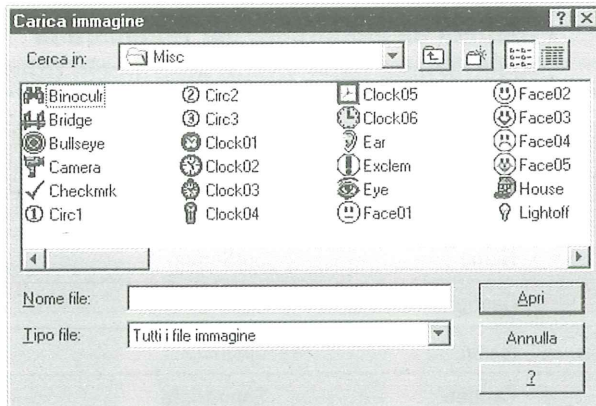
7. **Fate clic nella finestra Proprietà, fate clic sulla proprietà (Name) e digitate imgFelice.**

8. **Fate clic nella finestra Proprietà, fate clic sulla proprietà Picture e quindi sui tre puntini di sospensione (...).**

   Visual Basic visualizza la finestra di dialogo Carica immagine.

9. **Aprite la cartella Icons (si trova nella cartella Graphics, a sua volta contenuta nella cartella di Visual Basic) e poi aprite la cartella Misc all'interno della cartella Icons.**

   Visual Basic visualizza la finestra di dialogo Carica immagine, come mostra la Figura 3.7

Figura 3.7 La finestra di dialogo Carica immagine

10. **Fate doppio clic sull'icona FACE03.**

    Visual Basic visualizza una faccina sorridente all'interno del riquadro immagine.

11. **Fate clic sulla proprietà Visible e poi fate clic sulla freccia verso il basso e selezionate False.**

12. **Fate clic nella casella di testo Text1.**

13. **Fate clic sulla proprietà Border Style e poi fate clic sulla freccia verso il basso e selezionate 0-None.**

14. **Fate clic sulla proprietà (Name) e digitate txtFelice.**

15. **Fate doppio clic sulla proprietà Text e poi premete BACKSPACE per cancellare il valore della proprietà Text. In questo modo non apparirà alcun testo all'interno dell'oggetto.**

16. **Terminate la modifica delle proprietà per il resto degli oggetti in base alla Tabella 3.1.**

17. **Selezionate File-Salva progetto per salvare le modifiche apportate.**

Congratulazioni! Avete appena definito tutte le proprietà necessarie per la vostra prima interfaccia utente.

Lasciare che sia Visual Basic a creare automaticamente un'interfaccia utente

Se avete effettivamente provato a utilizzare la procedura per creare un'interfaccia utente partendo da zero, vi sarete sicuramente resi conto di quanto possa essere

Tabella 3.1 Proprietà da modificare per terminare la progettazione della vostra interfaccia utente

Oggetto	*Proprietà*	*Valore*
Form	Name	frmCiao
	Caption	Ciao a tutti!
Option2	Name	optSereno
	Caption	Sono sereno
Option3	Name	optTriste
	Caption	Sono triste
Image2	Name	imgSereno
	Picture	FACE02
	Visible	False
Image3	Name	ImgTriste
	Picture	FACE01
	Visible	False
Text2	Name	txtSereno
	Border Style	0-None
	Text	(Vuoto)
Text3	Name	txtTriste
	Border Style	0-None
	Text	(Vuoto)
Command1	Name	cmdEsci
	Caption	Esci

noiosa e lenta. Dato che i computer dovrebbero permettere di risparmiare tempo (in modo da averne di più per fare i giochi), Visual Basic offre la funzione Creazione guidata applicazioni VB.

Questa funzione permette di creare la struttura di un programma con i menu a discesa, le barre degli strumenti e le finestre di dialogo. Se volete creare un programma che richiede un'interfaccia utente simile a quella di Microsoft Word, Excel o PowerPoint, potete risparmiare tempo lasciando che sia la funzione Creazione guidata applicazioni VB a creare la vostra interfaccia utente.

Per vedere come funziona, seguite la procedura di seguito riportata.

1. **In Windows 95, 98, NT avviate Microsoft Visual Basic (se il programma è già in esecuzione, selezionate File-Nuovo progetto).**

   Visual Basic visualizza la finestra di dialogo Nuovo progetto.

2. **Fate clic sull'icona Creazione guidata applicazioni VB.**

   Verrà così visualizzata la finestra Creazione guidata applicazioni - Introduzione, come illustrato nella Figura 3.8. Questa finestra chiede di indicare quale profilo utilizzare. I *profili* sono impostazioni quadro che consentono di personalizzare la Creazione guidata applicazioni. Per il momento, accontentatevi dell'opzione (None).

3. **Fate clic su Avanti.**

   Appare la finestra Creazione guidata applicazioni - Tipo di interfaccia, come illustrato nella Figura 3.9.

4. **Fate clic sul pulsante di opzione Interfaccia a documenti multipli (MDI) e fate clic su Avanti.**

   Appare la finestra Creazione guidata applicazioni - Menu, come illustrato nella Figura 3.10. (Per scoprire come sono le Single Document (SDI), fate clic su un pulsante di opzione differente).

5. **Fate clic su Avanti per accettare la selezione di default.**

   Viene così visualizzata la finestra Creazione guidata applicazioni - Personalizza barra degli strumenti, presentata nella Figura 3.11. Trascinando e rilasciando le icone è possibile creare barre degli strumenti proprie.

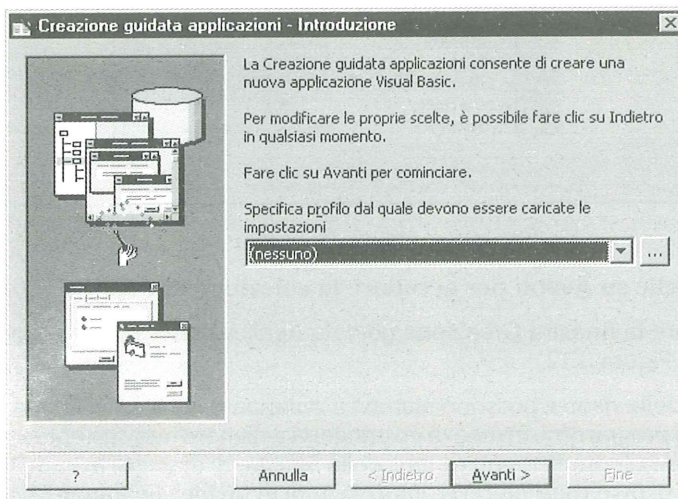



Figura 3.8 La finestra Creazione guidata applicazioni - Introduzione

Figura 3.9 La finestra Creazione guidata applicazioni - Tipo di interfaccia

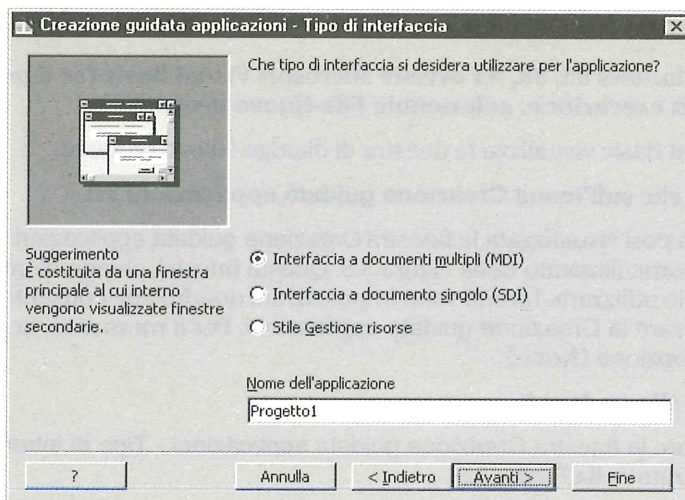
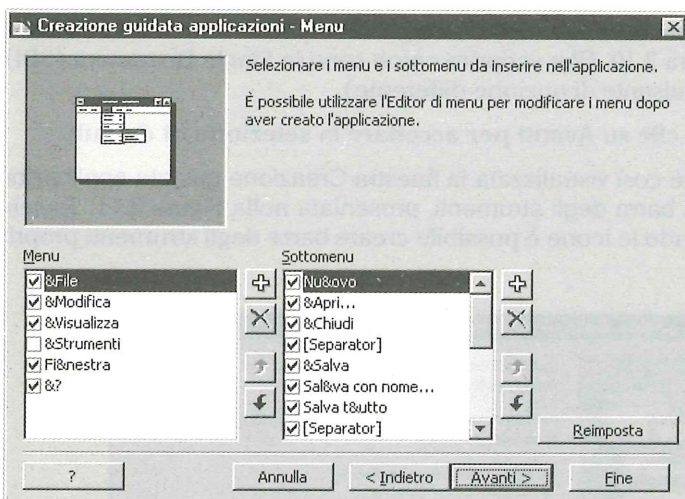



Figura 3.10 La finestra Creazione guidata applicazioni - Menu

6. **Fate clic su Avanti per accettare la selezione di default.**

   Appare la finestra Creazione guidata applicazioni - Risorse, come illustrato nella Figura 3.12.

   I file delle risorse possono aiutarvi a sviluppare versioni in lingua straniera dei vostri programmi. Invece di costringervi a digitare nuovamente il testo del programma (come i menu e i titoli delle finestre di dialogo) in varie lingue, un file delle risorse memorizza le informazioni in un file separato e vi consente così di utilizzare un file delle risorse diverso per ogni lingua desiderata.

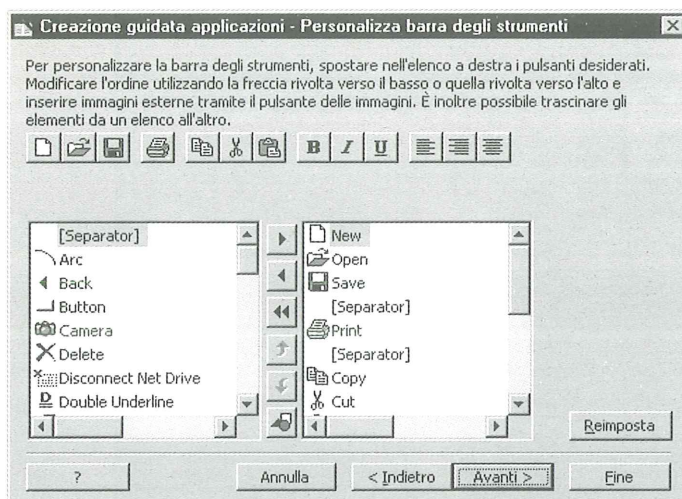



Figura 3.11 La finestra Creazione guidata applicazioni-Personalizza barra degli strumenti

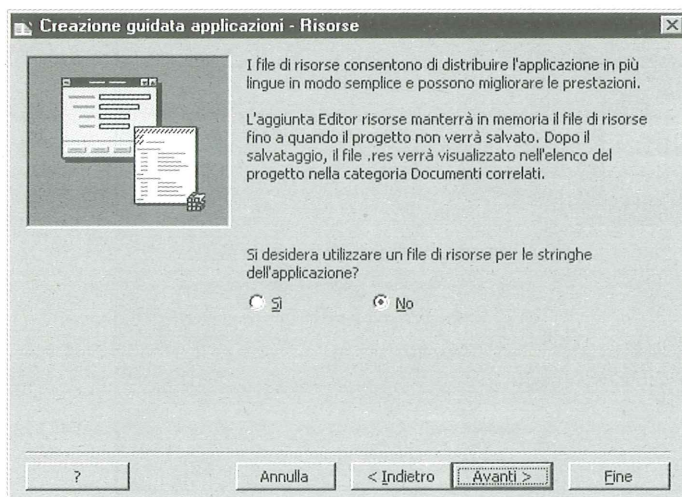



Figura 3.12 La finestra Creazione guidata applicazioni - Risorse

7. **Fate clic su Avanti per accettare la selezione di default.**

   Appare la finestra Creazione guidata applicazioni - Connessione a Internet, come illustrato nella Figura 3.13. In questo punto potete specificare se desiderate che il vostro programma sia in grado di accedere a Internet.

8. **Fate clic su Avanti per accettare la selezione di default.**

   Appare la finestra Creazione guidata applicazioni - Form standard, come illustrato nella Figura 3.14.

Figura 3.13 La finestra Creazione guidata applicazioni - Connessione a Internet

Creazione guidata applicazioni - Form standard

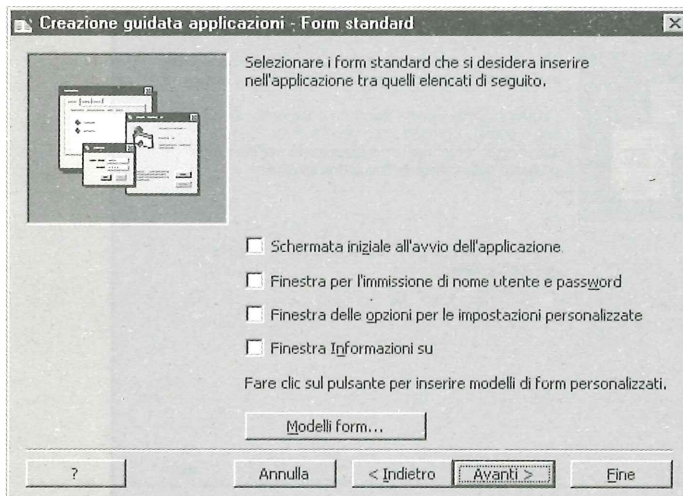
Selezionare i form standard che si desidera inserire nell'applicazione tra quelli elencati di seguito.

- Schermata iniziale all'avvio dell'applicazione
- Finestra per l'immissione di nome utente e password
- Finestra delle opzioni per le impostazioni personalizzate
- Finestra Informazioni su

Fare clic sul pulsante per inserire modelli di form personalizzati.

Modelli form...

? | Annulla | < Indietro | Avanti > | Fine

Figura 3.14 La finestra Creazione guidata applicazioni - Form standard

9. **Fate clic sulla casella di controllo Finestra Informazioni su e poi fate clic su Avanti.**

   Appare la finestra Creazione guidata applicazioni - Form per l'accesso ai dati, come illustrato nella Figura 3.15. In questo punto potete specificare il tipo di file di database che il programma deve utilizzare.

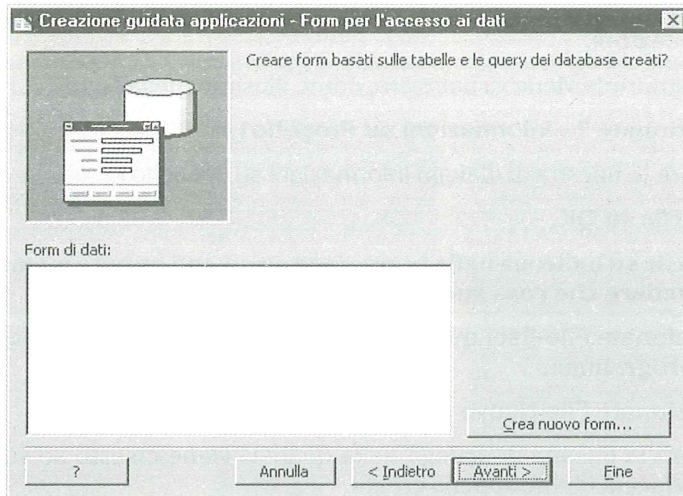



Figura 3.15 La finestra Creazione guidata applicazioni - Form per l'accesso ai dati

10. **Fate clic su Avanti per accettare la selezione di default.**

    Appare la finestra Creazione guidata applicazioni - Operazione completata, come illustrato nella Figura 3.16.

11. **Fate clic sul pulsante Fine.**

    Visual Basic visualizza la finestra di dialogo Applicazione creata.

12. **Fate clic su OK.**

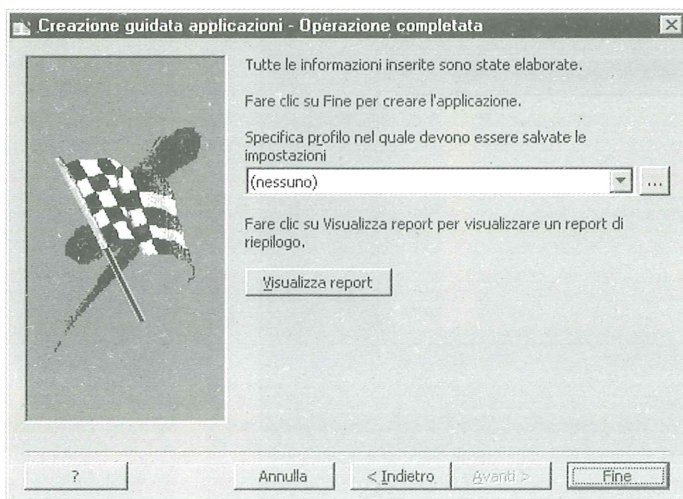



Figura 3.16 La finestra Creazione guidata applicazioni - Operazione completata

13. **Premete F5 per eseguire il programma creato dalla Creazione guidata applicazioni.**

    Il programma viene visualizzato, come illustrato nella Figura 3.17.

14. **Selezionate ? - Informazioni su Progetto1...**

    Appare la finestra di dialogo Informazioni su Progetto1.

15. **Fate clic su OK.**

16. **Fate clic su un'icona nella barra degli strumenti oppure su uno dei menu per vedere che cosa succede.**

17. **Selezionate File-Esci quando avete terminato di giocherellare con questo programma.**

18. **Selezionate File-Nuovo progetto.**

    Appare la finestra di dialogo nella quale vi viene chiesto se volete salvare l'interfaccia utente appena creata.

19. **Fate clic su No.**

Come potete vedere, la Creazione guidata applicazioni è in grado di creare un'interfaccia utente comprendente menu a discesa e barra degli strumenti. Tuttavia, se volete che il programma svolga qualche operazione degna di questo nome dovete personalizzare l'interfaccia utente e scrivere il codice BASIC. La funzione Creazione guidata applicazioni VB rappresenta semplicemente un vantaggio nella creazione della vostra interfaccia utente.

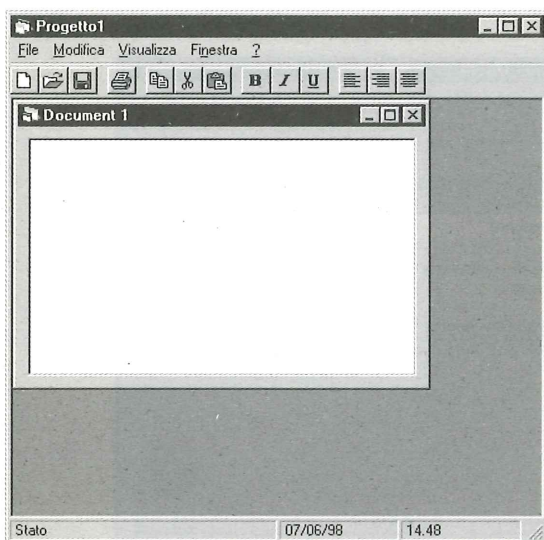



Figura 3.17 La struttura del programma creata dalla Creazione guidata applicazioni

Capitolo 4

Scrivere un codice BASIC

In questo capitolo

- I principi di base del codice BASIC
- Scegliere un oggetto e scrivere una procedura
- Scrivere il codice BASIC per il vostro programma

Affinché il vostro computer faccia qualcosa, dovrete fornirgli le istruzioni relative a ogni singolo passo. Se saltate un passo oppure gli impartite istruzioni poco chiare, il vostro computer non saprà che cosa fare (anzi, lo saprà, ma non farà quello che volevate voi).

- I programmatori definiscono ogni istruzione un *comando*. Un comando BASIC tipico potrebbe essere il seguente:

```
Tasse = Reddito * Aliquota fissa
```

- Una serie di comandi viene definita *codice*. Una serie tipica di comandi potrebbe essere la seguente:

```
Reddito = 90000
Aliquota fissa = 0,35
Tasse = Reddito * Aliquota fissa
```

- Il codice che costringe il vostro programma a fare qualcosa di utile (per esempio far partire un gioco, calcolare le tasse o visualizzare tostapane volanti sullo schermo) viene chiamato *programma*.

Se volete parlare la lingua dei programmatori, sebbene sia risaputo che i programmatori non dicono mai granché, dovrete imparare il galateo della programmazione.

Non si scrive mai un programma, si scrive il codice. Che il cielo vi perdoni se direte: "Fammi vedere la tua serie di comandi". I veri programmatori arrossiranno alla vostra gaffe. Dite invece: "Lasciami dare un'occhiata al tuo codice".

Che cos'è il codice Basic

Per fare in modo che il vostro computer faccia qualcosa, dovrete impartirgli istruzioni che esso sia in grado di comprendere. Dato che state utilizzando Visual Basic, dovrete utilizzare il linguaggio di programmazione BASIC.

Proprio come tutti gli altri linguaggi, il BASIC dispone di comandi speciali definiti *parole chiave riservate*.

Ecco alcuni esempi di parole chiave riservate:

Loop	Function	Sub	End
Do	Integer	Case	If
Else	Select	Then	For

Il codice BASIC è composto da una serie di parole chiave riservate BASIC collegate in maniera creativa per formare un programma. Ogni volta che il computer vede una parola chiave riservata, pensa automaticamente: "Ecco l'istruzione speciale alla quale so già come ubbidire".

Un programma può essere breve tanto quanto una singola parola chiave riservata oppure lungo e composto da milioni di parole chiave riservate. I programmi brevi di solito si limitano a visualizzare sullo schermo una frase tipo Ciao, Mondo!. Solitamente i programmi lunghi fanno qualcosa di più, ma spesso sono difficili da leggere quanto il modulo per pagare le tasse.

Dal punto di vista teorico, è possibile scrivere un unico programma lungo che comprenda un milione di parole chiave o anche di più. Tuttavia, qualsiasi programmatore che tenti una simile impresa ha altissime probabilità di impazzire prima di raggiungere lo scopo.

Scrivere un programma un passo alla volta

Per rendere la programmazione più facile, la maggior parte dei programmatori suddivide un programma grande in numerosi programmi più piccoli. Una volta terminato di scrivere i programmi piccoli è possibile riunirli in modo da formare un programma completo.

- ✓ Quando si suddivide un programma grande in numerosi programmi piccoli, questi ultimi vengono definiti *sottoprogrammi*. Nel linguaggio di Visual Basic, i sottoprogrammi vengono chiamati *procedure di evento* (sebbene alcuni programmatori parlino di *subroutine*). Visual Basic ha anche sottoprogram-

mi speciali definiti *funzioni* (illustrati nel Capitolo 29) e *procedure generali* (illustrati nel Capitolo 27).

- ✓ Le procedure comunicano a ogni oggetto presente nella form come reagire a un'azione dell'utente. Ciascun oggetto può disporre di zero o più procedure che gli dicono come reagire alle azioni dell'utente.

Una procedura può comunicare al computer che cosa fare se l'utente seleziona un oggetto mediante il mouse (come un pulsante di comando). Un'altra procedura può dire al computer come comportarsi se l'utente preme un determinato tasto mentre l'oggetto è evidenziato.

Le procedure non sono richieste da tutti gli oggetti, ma solo da quelli che l'utente seleziona o evidenzia in qualche modo, come i pulsanti di comando, le caselle di controllo o i pulsanti di opzione.

Scegliere oggetti ed eventi

Prima di poter scrivere una procedura per un oggetto, dovrete dire a Visual Basic i seguenti dati.

- ✓ Il nome dell'oggetto da utilizzare.
- ✓ L'evento al quale volete che l'oggetto risponda.

Visual Basic offre due metodi per selezionare un oggetto.

- ✓ Il metodo più semplice consiste nel fare clic su un oggetto (per esempio un pulsante di comando) della form e premere F7 (oppure fare doppio clic sull'oggetto). Appare in questo modo la finestra codice, all'interno della quale Visual Basic inserisce automaticamente la prima e l'ultima riga della procedura relativa all'evento di default per l'oggetto selezionato.
- ✓ Il secondo metodo è quasi altrettanto semplice. Ogni volta che disegnate un oggetto su una form, Visual Basic memorizza il nome dell'oggetto nell'elenco degli oggetti situato nella parte superiore sinistra della finestra codice, come illustrato nella Figura 4.1. Per poter scorrere tale elenco e scegliere l'oggetto per il quale si desidera scrivere una procedura, è necessario aprire la finestra codice selezionando Visualizza-Codice. Scelto l'oggetto, Visual Basic inserirà automaticamente la prima e l'ultima riga della procedura relativa all'evento di default.

Per selezionare un evento da utilizzare, fate clic sulla lista Procedure, che appare in cima alla finestra Code accanto alla lista Object. La lista Procedure contiene tutti gli eventi ai quali la procedura che scriverete potrà rispondere.

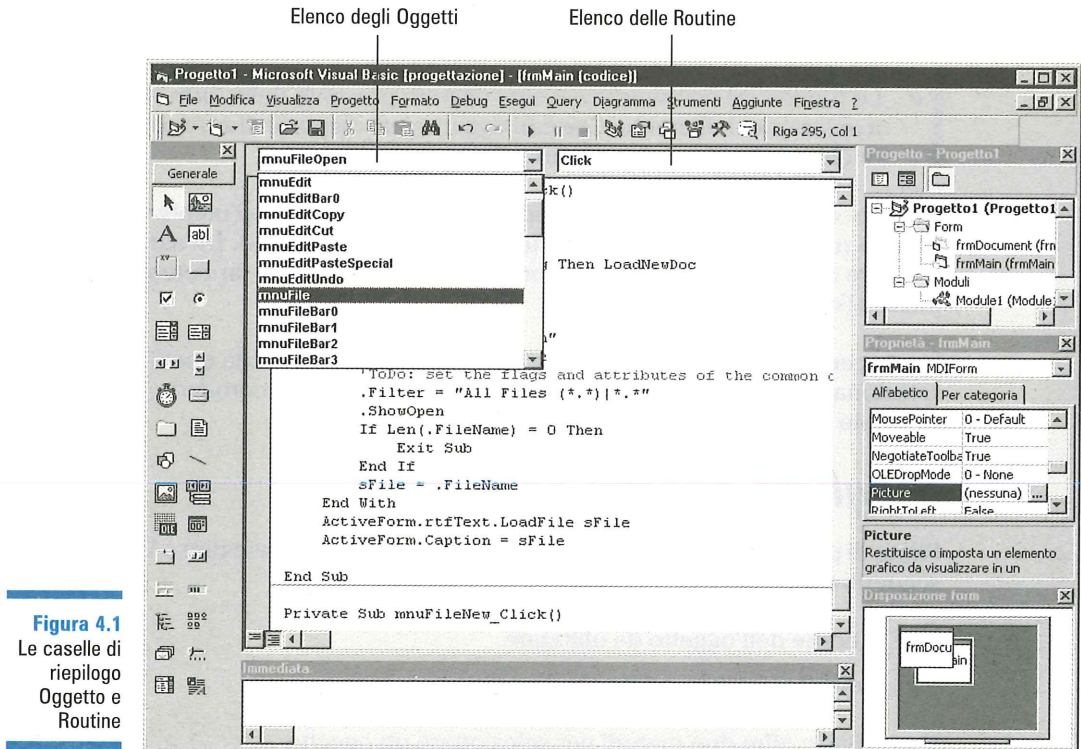



Figura 4.1 Le caselle di riepilogo Oggetto e Routine

Scorrete questo elenco fino a quando trovate l'evento che avete bisogno per scrivere un procedura. L'evento più usato è Click event.

Dopo aver selezionato un oggetto dalla lista Object, e un evento dalla lista Procedure, Visual Basic inserisce la prima e l'ultima riga della procedura. Ora siete pronti per iniziare a scrivere il codice per questa procedura!

Scrivere le procedure di evento in Visual Basic

Prima che possiate scrivere una procedura in Visual Basic dovete disegnare qualche oggetto sulla form.

Dopodiché, dovete modificare le proprietà di ogni oggetto in modo da attribuire a ciascuno di essi un nome univoco che sia facile da ricordare. In caso contrario, sarete obbligati a impiegare i nomi generici che Visual Basic fornisce di default per qualsiasi cosa, quali Option1 o Text3.

Sapete che cosa succede se scrivete una procedura per un particolare oggetto e poi modificate il nome dell'oggetto? Visual Basic si confonde e pensa che abbiate creato un oggetto nuovo di zecca, il che significa che all'oggetto rinominato non risulta associata alcuna procedura. Quindi, se avete intenzione di rinominare un oggetto fatelo prima di scrivere la procedura relativa a tale oggetto.

Per scrivere una procedura in Visual Basic per un determinato oggetto, aprite la finestra codice utilizzando uno dei tre metodi: fate clic sull'oggetto e premete F7, selezionate Visualizza-Codice e scegliete l'oggetto dall'elenco Oggetti, oppure fate doppio clic sull'oggetto.

Una volta visualizzata la finestra codice potete iniziare a digitare la procedura, o meglio, il codice della procedura. Nella Figura 4.2 è illustrata la finestra codice.

Aspettate! Visual Basic non si limita a visualizzare una finestra vuota, ma digita automaticamente `Private Sub`, seguito dal nome dell'oggetto, da un trattino di sottolineatura, da un evento (per esempio `Click`) e da due parentesi `()`, che potrebbero essere vuote oppure potrebbero racchiudere alcuni dati (questo argomento verrà affrontato in modo più dettagliato nel Capitolo 28).

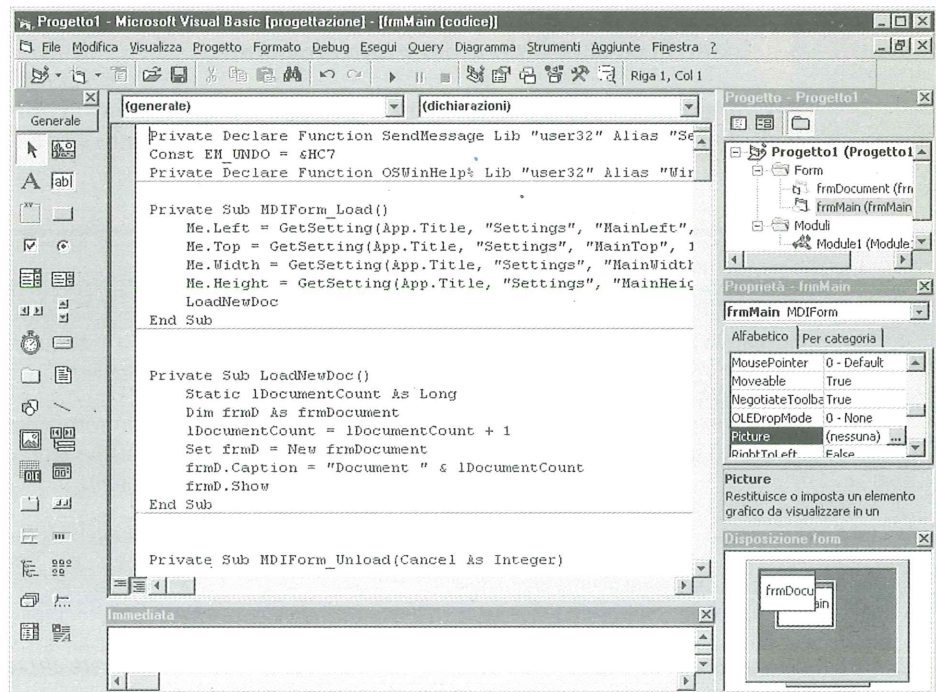



Figura 4.2 Una tipica vista della finestra Codice, dove appaiono varie procedure per gli eventi

Se fate doppio clic sull'oggetto denominato cmdEsci, Visual Basic visualizza quanto segue:

```
Private Sub cmdExit_Click()
End Sub
```

- ✓ La prima riga di una procedura Visual Basic comincia con `Private Sub`. Ciò significa che in questo punto inizia una procedura, o sottoprogramma, che appartiene a un unico oggetto specifico. In questo caso il sottoprogramma appartiene all'oggetto denominato `cmdExit`.
- ✓ Visual Basic inserisce quindi il nome dell'oggetto. Se vi siete dimenticati di modificare la proprietà del nome dell'oggetto, Visual Basic utilizza il nome di default come `Button1`. In caso contrario, Visual Basic visualizza il nome dell'oggetto (per esempio, `cmdExit`).
- ✓ Il nome dell'oggetto è seguito da un trattino di sottolineatura che separa il nome dell'oggetto dall'evento. Un *evento* è un'azione che l'utente effettua per comunicare con il computer. In questo esempio, l'evento è Click e questo significa che l'utente ha fatto clic con il mouse sull'oggetto chiamato cmdExit.
- ✓ Subito dopo vengono le due parentesi vuote. Talvolta le parentesi contengono i dati che il sottoprogramma utilizza quando viene eseguito. Due parentesi vuote indicano che questo sottoprogramma non richiede dati speciali da parte di un'altra sezione del vostro programma.

Per tradurre questa procedura in un linguaggio comprensibile, la prima riga significa: "Questo è un sottoprogramma per l'oggetto denominato `cmdExit` e il sotto-

Mettete alla prova le vostre nuove conoscenze

1. Che cosa sono le parole chiave riservate?

a. Parole da utilizzare per riservare un tavolo in uno splendido ristorante dove un bicchiere d'acqua costa 10.000 lire.

b. Quello che le persone timide vorrebbero poter dire.

c. Istruzioni speciali presenti in ogni linguaggio di programmazione.

d. Parole che vorreste poter dire davanti a qualcuno che vi sta antipatico.

2. Come si fa a scrivere un programma di ampie dimensioni senza impazzire?

a. Si divide il programma in sottoprogrammi, che Visual Basic definisce anche procedure (come eventi o procedure generali).

b. Si spiano gli altri che scrivono il programma e poi si ruba il loro lavoro.

c. Devo davvero scrivere un programma?

d. Se state meditando di scrivere un programma di ampie dimensioni, allora forse siete già impazziti.

programma comunica al computer che cosa fare se l'utente fa clic sull'oggetto cmdEsci".

L'ultima riga di questa procedura Visual Basic è composta da due parole: `End` e `Sub` e dice al computer: " Questa è la fine di tutti i comandi che appartengono a questo sottoprogramma". Invece di digitare tutte queste parole, Visual Basic utilizza semplicemente:

```
End Sub
```

Al momento, questa procedura di Visual Basic non fa nulla. Perché la procedura faccia qualcosa, dovete aggiungere i comandi tra la prima e l'ultima riga. Prima di cominciare ad aggiungere i comandi, dovrete sapere a cosa servono i comandi (vale a dire il codice) BASIC.

Che cosa può fare il codice Basic

Il codice BASIC può fare quanto segue.

- ✓ Calcolare un risultato.
- ✓ Modificare le proprietà (l'aspetto) di un oggetto.

Se volete calcolare il numero di persone che vivono su una barca a vela, che si sono abbonate al *National Geographic* e che possiedono un gatto, Visual Basic è in grado di farlo a patto che gli forniate tutti i dati necessari.

Dopo avere calcolato un risultato, con ogni probabilità vorrete visualizzarlo sullo schermo. A questo scopo dovrete modificare le proprietà di un oggetto presente sulla vostra interfaccia utente. Per esempio, se intendete visualizzare un messaggio sullo schermo, dovete innanzitutto disegnare su una form Visual Basic un oggetto rappresentato da una casella di testo.

Dopodiché dovrete attribuire a questa casella di testo un nome tipo txtMessaggio. Infine, per visualizzare qualcosa in questa casella di testo, dovete modificare la proprietà Text della casella di testo txtMessaggio in questo modo:

```
txtMessaggio.Text = "Questo è difficile da spiegare"
```

Questo comando visualizza il messaggio `Questo è difficile da spiegare` nella casella di testo txtMessaggio visualizzata sullo schermo, come illustrato nella Figura 4.3.

Ovviamente, il codice Visual Basic non può modificare tutte le proprietà di un oggetto. Alcune proprietà (come il nome dell'oggetto) possono essere modificate soltanto durante la fase di progettazione utilizzando la finestra delle proprietà dell'oggetto stesso.

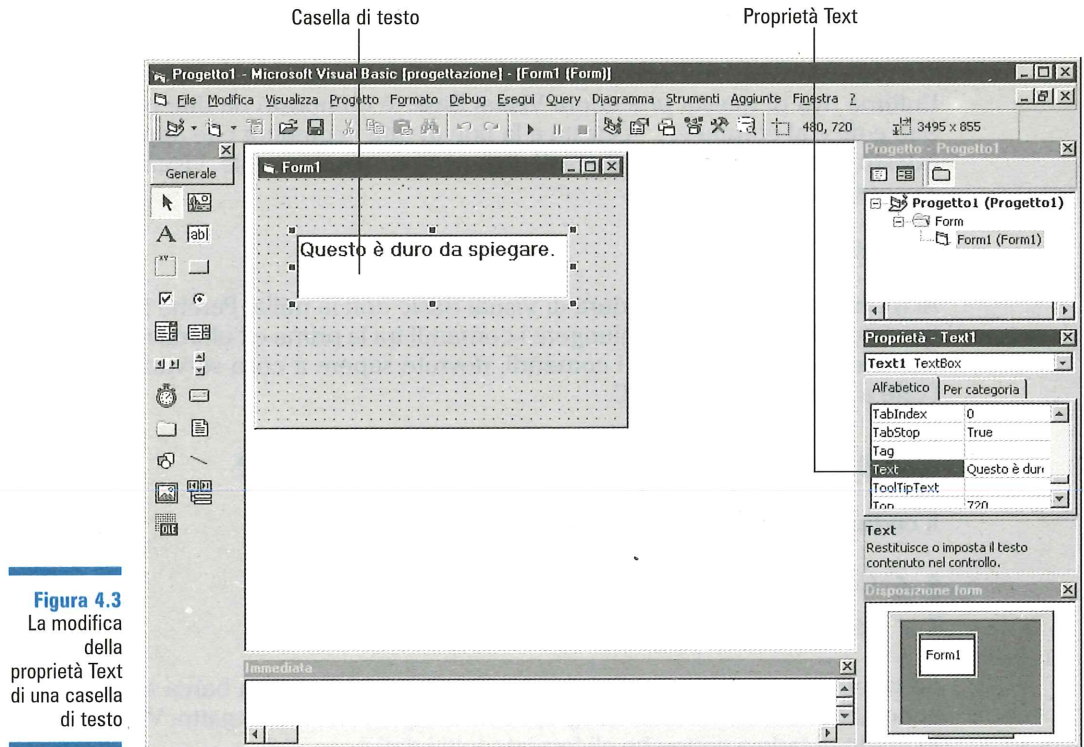



Figura 4.3 La modifica della proprietà Text di una casella di testo

Come funziona una procedura Visual Basic

In Visual Basic le istruzioni in una procedura vengono eseguite solo quando si verifica un evento specifico, per esempio quando l'utente fa clic su un oggetto. Lo stesso insieme di istruzioni viene ripetuto ogni volta che l'utente fa clic su un oggetto. Un programma Visual Basic giunge al termine solo quando la procedura di un oggetto comunica specificatamente al programma di concludere.

Osservate per esempio la procedura Visual Basic più semplice, necessaria affinché qualsiasi programma si fermi completamente. Per il semplice programma "Ciao a tutti!", la procedura necessaria sarebbe la seguente:

```
Private Sub cmdExit_Click()
     Unload Me
End Sub
```

Il codice di questa procedura contiene una parola chiave riservata denominata Unload e una variabile definita Me. La variabile Me rappresenta la form, quindi

questa procedura comunica a Visual Basic: "Scarica la form attualmente visualizzata dalla memoria". Dato che questo programma consiste di un'unica form, il comando interrompe effettivamente l'esecuzione del programma.

Questa procedura viene eseguita solo quando l'utente fa clic sull'oggetto cmdExit. Se osservate l'oggetto cmdExit sulla vostra interfaccia utente, vedrete che si tratta di un pulsante di comando sul quale appare l'etichetta Esci.

Un'alternativa al comando Unload Me per terminare un programma Visual Basic consiste nell'utilizzare il comando End, come indicato qui di seguito.

```
Private Sub cmdExit_Click()
  End
End Sub
```

Sì, ma qual è il metodo migliore? Il comando End costringe il programma a terminare immediatamente l'esecuzione, il che può equivalere a fermare l'auto portandola a sbattere contro un muro. Per concludere il vostro programma adottando un approccio più dolce e gentile, Microsoft raccomanda di utilizzare il comando Unload Me.

Ecco che cosa accade se eseguite questo programma.

1. Visual Basic visualizza sullo schermo la vostra interfaccia utente, compreso il pulsante di comando cmdExit. Questo nome tuttavia non appare sull'interfaccia utente dove, invece, viene visualizzata la proprietà Caption del pulsante di comando vale a dire la parola Esci.
2. Facendo clic sul pulsante Esci Visual Basic chiede: "Come si chiama questo oggetto appena selezionato dall'utente?"
3. In un batter d'occhio, Visual Basic nota che il nome del pulsante Esci è cmdExit.
4. Poi Visual Basic chiede: "Ci sono istruzioni che mi dicono che cosa fare se l'utente fa clic sull'oggetto cmdExit?". Fortunatamente Visual Basic trova la procedura `Private Sub cmdExit_Click()`.
5. Visual Basic esamina quindi la prima istruzione della procedura `cmdExit_Click()`. In questo caso, l'istruzione è `Unload Me`, che comunica a Visual Basic di scaricare la form. Dato che questa è l'unica form nel programma, questa istruzione permette effettivamente di interrompere il programma.
6. Visual Basic interrompe l'esecuzione del programma e lo cancella dallo schermo. Ovviamente, tutto questo succede in un batter d'occhio e il computer sembra reagire all'istante.

Scrivere il codice Basic per il vostro primo programma in Visual Basic

Coloro che amano tuffarsi nel programma senza dover digitare tutto il codice BASIC, possono scaricare il programma HELLO.VBP dal CD-ROM accluso.

Dato che l'esperienza è sempre la migliore maestra, la procedura di seguito riportata illustrerà come scrivere un vero codice BASIC che potrete utilizzare per fare colpo sui vostri amici.

Non preoccupatevi di capire tutto quello che digitate.

Lo scopo di questa esercitazione è solo quello di mostrarvi come sia semplice creare un programma in Visual Basic.

1. **Avviate Microsoft Visual Basic, sempre che non l'abbiate già fatto (oppure selezionate File - Apri progetto).**

   Visual Basic visualizza la finestra di dialogo Nuovo progetto (oppure la finestra Apri progetto).

2. **Fate clic sulla scheda Recente, fate clic su CIAO e poi su Apri.**

   Se la form CIAO.FRM non appare sullo schermo, fate clic sul nome della form, frmCiao, nella finestra Progetto e poi selezionate l'icona Visualizza oggetto. Questa operazione può essere saltata se la form in questione è già visualizzata sullo schermo dal Capitolo 3.

3. **Fate clic sul pulsante di opzione optFelice visualizzato nell'angolo superiore sinistro della form (è il pulsante che dice "Sono felice!").**

4. **Aprite la finestra codice selezionando Visualizza-Codice.**

5. **Digitate la procedura `Private Sub optFelice_Click()` in modo che il risultato sia il seguente:**

```
Private Sub optFelice_Click()
     imgFelice.Visible = True
     imgSereno.Visible = False
     imgTriste.Visible = False
     txtFelice. TEXT = "Ciao a tutti!"
     txtSereno.TEXT = ""
     txtTriste.TEXT = ""
End Sub
```

6. **Fate clic sulla freccia verso il basso nella casella di riepilogo Oggetto nella parte superiore della finestra Codice e poi selezionate l'oggetto optSereno.**

Visual Basic visualizza una procedura vuota `Private Sub optSereno_Click()`.

7. **Completate la procedura `Private Sub optSereno_Click()` in modo che il risultato sia il seguente:**

```
Private Sub optSereno_Click()
     imgFelice.Visible = False
     imgSereno.Visible = True
     imgGrin.Triste = False
     txtFelice. TEXT = ""
     txtSereno.TEXT = "Oggi è una bella giornata."
     txtTriste.TEXT = ""
End Sub
```

8. **Fate clic sulla freccia verso il basso nella casella di riepilogo Oggetto nella parte superiore della finestra Codice e poi selezionate l'oggetto optTriste.**

   Visual Basic visualizza una procedura vuota `Private Sub optTriste _Click()`.

9. **Completate la procedura `Private Sub optTriste_Click()` in modo che il risultato sia il seguente:**

```
Private Sub optTriste_Click()
     imgFelice.Visible = False
     imgSereno.Visible = False
     imgTriste.Visible = True
     txtFelice. TEXT = ""
     txtSereno.TEXT = ""
     txtTriste.TEXT = "Addio mondo crudele..."
End Sub
```

10. **Fate clic sull'elenco Oggetto nella parte superiore della finestra Codice e poi selezionate l'oggetto cmdEsci.**

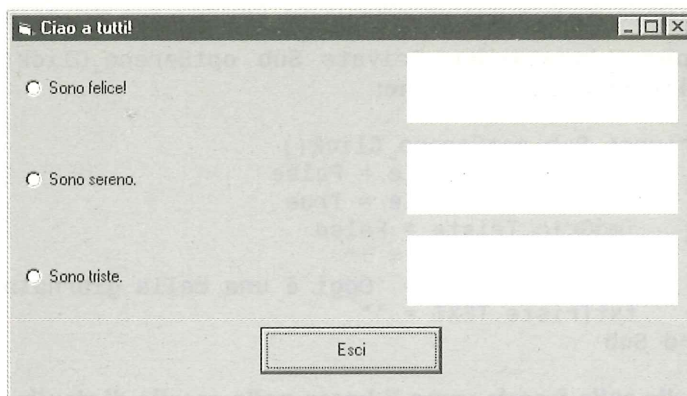
    Visual Basic visualizza una procedura vuota `Private Sub cmdEsci_Click()`.

11. **Completate la procedura `Private Sub cmdEsci_Click()` in modo che il risultato sia il seguente:**

```
Private Sub cmdEsci_Click()
     Unload Me
End Sub
```

12. **Per eseguire il programma premete F5 oppure selezionate Esegui-Avvia.**

    Se avete digitato tutto correttamente Visual Basic visualizza la vostra interfaccia utente sullo schermo, come illustrato nella Figura 4.4.

Figura 4.4 Il programma Ciao a tutti!

13. **Fate clic sul pulsante di opzione accanto alla dicitura "Sono triste".**

    Visual Basic visualizza sullo schermo una faccina accompagnata dal messaggio "Addio mondo crudele...", come illustrato nella Figura 4.5.

14. **Fate clic sul pulsante di opzione accanto alla dicitura "Sono sereno".**

    Visual Basic visualizza sullo schermo una faccina sorridente accompagnata dal messaggio "Oggi è una bella giornata", come illustrato nella Figura 4.6.

15. **Fate clic sul pulsante di opzione accanto alla dicitura "Sono felice".**

    Visual Basic visualizza sullo schermo una faccina davvero felice accompagnata dal messaggio "Ciao a tutti", come illustrato nella Figura 4.7.

16. **Fate clic sul pulsante di comando Esci.**

    Visual Basic interrompe l'esecuzione del programma e ritorna a Visual Basic.

Avete finalmente completato l'esempio Ciao a tutti! A questo punto, avete imparato a utilizzare Visual Basic per creare una gradevole interfaccia utente in modo rapido e facile.

Figura 4.5
Addio mondo crudele

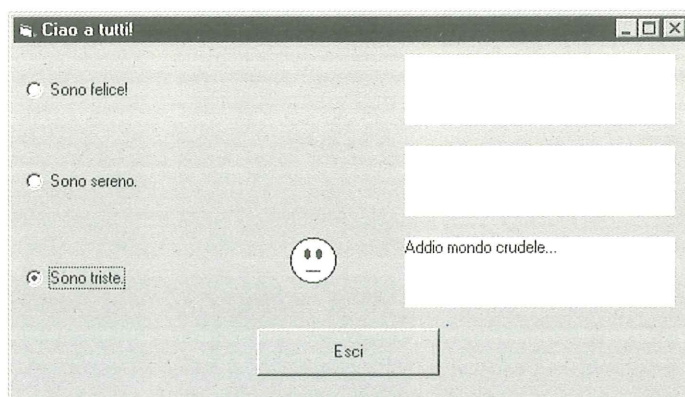



Figura 4.6
Oggi è una bella giornata

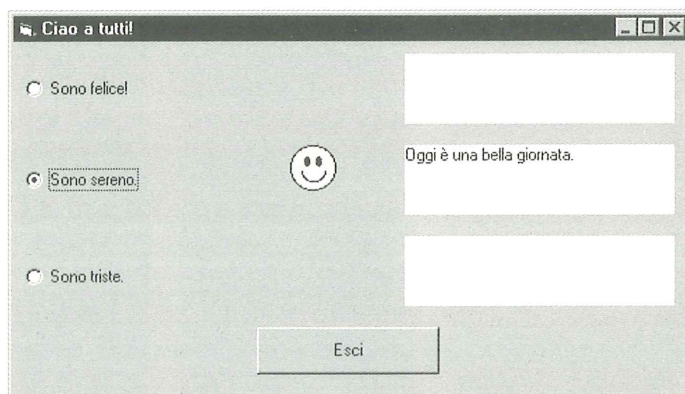
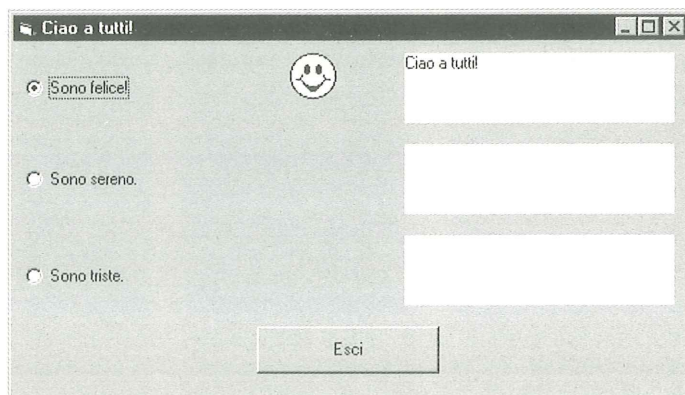



Figura 4.7
Ciao a tutti!

Parte II

Creare le interfacce utente

I VERI PROGRAMMATORI ODIANO ESSERE DISTURBATI
MENTRE STANNO LAVORANDO

In questa parte...

La funzione di un'interfaccia utente è consentire a qualcuno di servirsi del vostro programma. Più l'interfaccia utente è confusa, più il vostro programma sarà difficile da utilizzare. Se riuscirete a creare un'interfaccia utente logica e di facile utilizzo, sicuramente il vostro programma verrà adottato da un gran numero di persone.

Questa è la parte divertente del libro. Non dovrete scrivere del codice bizzarro, imparare strani comandi o imparare a memoria tasti misteriosi. Potrete scarabocchiare sullo schermo del computer; in realtà starete scoprendo come scrivere i vostri programmi.

Capitolo 5

Struttura dell'interfaccia utente

In questo capitolo

- Alcuni consigli per creare un'interfaccia utente
- Disegnare oggetti sull'interfaccia utente
- Spostare, cancellare e copiare oggetti
- Disattivare gli oggetti

Per farsi capire dagli altri, bisogna innanzitutto parlare ai loro occhi.

Napoleone Bonaparte

Tanto per cominciare, sappiate che nessuno vuole davvero utilizzare il vostro programma. La maggior parte delle persone preferisce stare sulla spiaggia, guardare la TV o andare a divertirsi. Tuttavia, potrebbero tornare utili a molti i risultati che il vostro programma può produrre. Se questi potessero ottenere gli stessi risultati con altri sistemi e meno lavoro, lo farebbero. Ma dato che ciò non è possibile, sono ben disposti a utilizzare il vostro programma.

È un fatto, la gente vorrebbe che il vostro programma potesse leggere nelle menti e poi, come per magia, facesse tutto il lavoro automaticamente. Dato che questo è impossibile, il meglio che possiate fare è rendere il programma davvero semplicissimo da utilizzare. Se un imbecille del tutto incompetente (il vostro capo) riesce a servirsi del programma, allora ci riuscirà anche la maggior parte delle altre persone.

Prima di creare l'interfaccia utente

Creare un'interfaccia utente non significa semplicemente mettere insieme qualche bella immagine su una finestra a colori e sperare che l'utente sia in grado di capire come funziona il programma. L'interfaccia utente deve rendere semplice l'uso del programma. Per aiutarvi a creare un'interfaccia utente, ecco alcuni punti che dovete tenere presente.

Conoscere l'utente

Prima di progettare l'interfaccia utente, chiedetevi chi utilizzerà il vostro programma. I vostri utenti tipici saranno impiegati addetti all'inserimento dati che conoscono i computer oppure dirigenti che capiscono solo le procedure su carta e non hanno la minima idea di come i computer possano aiutarli?

Quando avrete individuato coloro che utilizzeranno il vostro lavoro, progettate un'interfaccia utente che rispecchi il modo in cui gli utenti già lavorano, senza preoccuparvi se risulta del tutto inefficiente o incomprensibile per gli altri. I contabili accettano di buon grado i fogli elettronici perché le righe e le colonne assomigliano ai fogli dei loro registri, mentre i dattilografi preferiscono i programmi di elaborazione testi, dato che ricordano un foglio di carta bianco.

Tuttavia, immaginate che cosa accadrebbe se tutti i programmi di elaborazione testi assomigliassero a fogli elettronici con righe e colonne. Qualsiasi dattilografa che cercasse di utilizzare questo tipo di programma si sentirebbe ben presto persa e confusa (mentre i contabili potrebbero sentirsi subito a loro agio).

Più un programmatore conosce l'utente, maggiori sono le probabilità che l'interfaccia venga utilizzata e accettata. L'unica persona che l'interfaccia utente deve realmente soddisfare è l'utente.

Orientare l'utente

Non c'è nulla di strano nel fatto che la gente si perda nei centri commerciali, disposti su più piani e con due fusi orari diversi. Come ci si può sentire quando non si ha idea di dove ci si trovi e non si sa dove si possa andare partendo da dove ci si trova?

Questa sensazione di smarrimento spiega perché tanti bambini piangano in maniera incontrollata e altrettanti utenti di computer imprechino sottovoce. Ecco perché nei centri commerciali ci sono grandi piantine con grosse X rosse che dicono: "Voi siete qui".

Un'interfaccia utente valida deve essere in grado di orientare la gente, in modo che sappia a che punto si trova all'interno del programma e come uscire nel caso desideri farlo. Alcune interfacce utente visualizzano un messaggio nella parte inferiore dello schermo, come `Pagina 2 di 5`. In questo caso l'utente sa esattamente quante pagine può visualizzare e quale pagina è presente in quel momento sullo schermo.

La vostra interfaccia utente rappresenta la mappa del programma; accertatevi che fornisca una quantità sufficiente di informazioni per orientare l'utente, ma non troppe, per evitare di confonderlo.

Rendere le scelte evidenti

Oltre a far sapere agli utenti in che punto del programma si trovano, una buona interfaccia utente deve anche rendere le scelte evidenti. Se l'interfaccia utente visualizza il messaggio `Pagina 4 di 25` nella parte inferiore dello schermo, come farà l'utente a sapere come fare a passare alla pagina seguente o a quella precedente? Una soluzione può essere quella di mostrare frecce rivolte a destra e a sinistra in ciascun angolo inferiore della pagina. Un'altra soluzione può essere quella di inserire pulsanti con la dicitura Pagina Precedente e Pagina successiva. Nella Figura 5.1 sono illustrate alcune possibili soluzioni.

Se il vostro programma mostrerà quali azioni sono disponibili e quali tasti premere o dove puntare il mouse, quando utilizzerà il programma l'utente avrà una sensazione di controllo e di sicurezza.

Siate clementi

Il trucco sta nel feedback. Se ogni volta che l'utente preme il tasto sbagliato o fa clic sull'area errata, il vostro programma assume un atteggiamento arrogante e visualizza messaggi come `File MRP.DLL non trovato`, con ogni probabilità l'utente si sentirà intimidito se il programma non gli spiega che cosa significa il messaggio di errore e come evitarlo.

Quindi siate gentili. Fate in modo che il programma nasconda o ombreggi tutti i tasti o i comandi di menu non utilizzabili. Qualora l'utente premesse un tasto sbagliato o facesse clic con il mouse nell'area errata, fate in modo che il programma visualizzi una finestra e spieghi all'utente quali siano le opzioni disponibili. Gli utenti apprezzeranno un programma capace di guidarli, il che significa che voi trascorrerete molto meno tempo a rispondere a telefonate che richiedono assistenza tecnica.

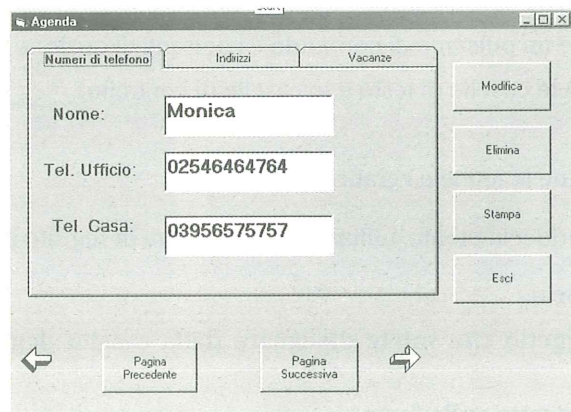



Figura 5.1 Offrire delle scelte evidenti sull'interfaccia utente

Viva la semplicità

La maggior parte dei programmi offre agli utenti due o più metodi per scegliere un comando specifico. È possibile fare clic su un pulsante, selezionare un comando dai menu oppure premere determinate combinazioni di tasti (per esempio, CTRL+F2). Di questi tre metodi, quello di fare clic su un pulsante rappresenta la procedura più semplice da ricordare, mentre premere strane combinazioni di tasti è sicuramente la più difficile.

Assicuratevi che i comandi utilizzati con maggiore frequenza possano essere raggiunti velocemente mediante un pulsante o un menu; non tutti i comandi devono essere necessariamente impartiti mediante una combinazione di tasti.

Sebbene le combinazioni di tasti siano più rapide da utilizzare, all'inizio sono anche molto difficili da imparare. Fate in modo che le combinazioni di tasti siano semplici da ricordare, quando è possibile. Per il comando Salva, CTRL+S risulterà più semplice da ricordare rispetto a una combinazione di tasti del tutto astratta come MAIUSC+F12. È facile ricordare la lettera S significa Salva, ma che cosa significa F12?

Progettare l'interfaccia utente in Visual Basic

Quando scrivete un programma Visual Basic, per prima cosa dovete progettare l'interfaccia utente. In linea di principio, un'interfaccia utente Visual Basic consiste di oggetti che voi disponete sullo schermo con una parvenza di organizzazione affinché la schermata abbia un aspetto piacevole. Gli elementi comuni di un'interfaccia utente Visual Basic (riportati nella Figura 5.2) comprendono quanto segue.

- ✓ Form (note anche come finestre)
- ✓ Pulsanti (come un pulsante di comando e i pulsanti di opzione)
- ✓ Caselle (come le caselle di testo e le caselle di controllo)
- ✓ Etichette
- ✓ Immagini (come le icone e i grafici).

Per progettare l'interfaccia utente, utilizzate la procedura di seguito illustrata.

1. **Create una form.**
2. **Scegliete l'oggetto che volete disegnare dalla casella degli strumenti (Tabella 5.1).**
3. **Disegnate l'oggetto sulla form.**

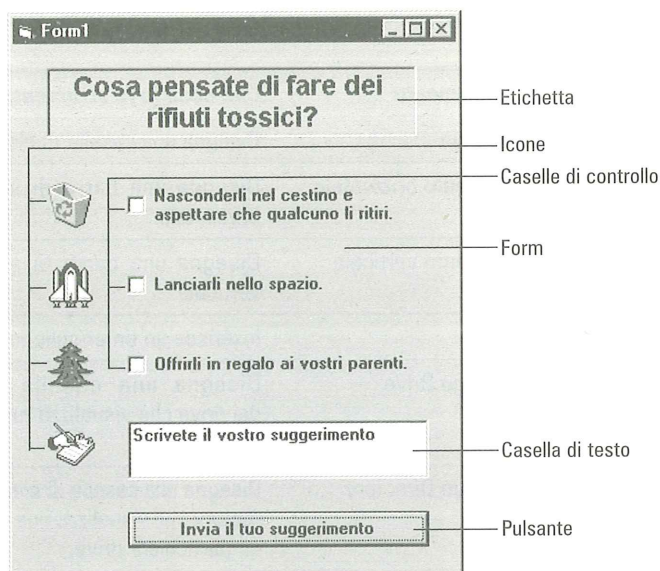



Figura 5.2 I componenti di una tipica interfaccia utente di Visual Basic

Tabella 5.1 Strumenti offerti dalla barra degli strumenti di Visual Basic

Icona	*Nome dello strumento*	*Che cosa fa lo strumento*
	Puntatore	Seleziona gli oggetti.
	Riquadro immagine	Disegna un riquadro per visualizzare immagini grafiche.
A	Etichetta	Disegna un riquadro per visualizzare un testo.
abl	Casella di testo	Disegna una casella che può visualizzare un testo e permettere all'utente di digitare un testo.
xv	Cornice	Raggruppa due o più oggetti insieme.
	Pulsante di comando	Disegna un pulsante di comando.
	Casella di controllo	Disegna una casella di controllo.
	Pulsante di opzione	Disegna un pulsante di opzione.
	Casella combinata	Disegna una casella combinata.

(continua)

Tabella 5.1

Icona	*Nome dello strumento*	*Che cosa fa lo strumento*
	Casella di riepilogo	Disegna una casella di elenco.
	Barra di scorrimento orizzontale	Disegna una barra di scorrimento orizzontale.
	Barra di scorrimento verticale	Disegna una barra di scorrimento verticale.
	Timer	Inserisce un timer sulla form.
	Casella di riepilogo Drive	Disegna una casella di elenco dei drive che visualizza tutti i drive disponibili.
	Casella di riepilogo Directory	Disegna una casella di elenco delle directory che visualizza una directory su un particolare drive.
	Casella di riepilogo File	Disegna una casella di elenco dei file che visualizza una directory specifica.
	Forma	Disegna una forma geometrica come un cerchio o un quadrato.
	Riga	Disegna una riga.
	Casella Immagine	Disegna una casella destinata a visualizzare immagini grafiche.
	Controllo dei dati	Disegna un controllo avente lo scopo di collegare un programma al file di un database.
	OLE	Disegna una casella per l'inserimento di un oggetto OLE.

Disegnare un oggetto

Per disegnare un oggetto, utilizzate la procedura di seguito illustrata.

1. **Fate clic sull'icona della casella degli strumenti di Visual Basic che rappresenta l'oggetto desiderato (un pulsante di comando, un riquadro di immagine, un'etichetta e così via).**
2. **Posizionate il puntatore del mouse all'interno della form, sul punto in cui desiderate disegnare l'oggetto.**

   Il cursore si trasforma in una croce.

3. **Fate clic e trascinate il mouse per definire le dimensioni del nuovo oggetto e poi rilasciate il pulsante del mouse.**

Se volete disegnare velocemente un oggetto su una form, fate doppio clic sull'icona dell'oggetto nella casella degli strumenti. Per esempio, se volete disegnare in fretta un pulsante di comando, fate semplicemente doppio clic sull'icona del pulsante di comando nella casella degli strumenti e Visual Basic disegnerà automaticamente il pulsante di comando.

Utilizzare la finestra Proprietà

Dopo avere disegnato un oggetto su una form, occorre definire le proprietà dell'oggetto. A questo scopo, dovete utilizzare la finestra Proprietà. Nel caso tale finestra non sia visualizzata sullo schermo, eseguite uno dei seguenti tre comandi per farla apparire.

- ✓ Premete F4.
- ✓ Selezionate Visualizza - Finestra Proprietà.
- ✓ Fate clic sull'icona Finestra Proprietà nella barra degli strumenti.

La finestra Proprietà può visualizzare le proprietà in due modi, (Figura 5.3).

- ✓ **Alfabetico.** Elenca le proprietà di un oggetto in ordine alfabetico, dalla A alla Z.
- ✓ **Per categoria.** Organizza le proprietà in categorie, come le proprietà che influiscono sull'aspetto, il comportamento o la posizione di un oggetto.

Dopo avere aperto la finestra Proprietà, utilizzate la procedura di seguito riportata per visualizzare le proprietà di un oggetto nel programma Visual Basic.

1. **Scegliete uno di questi metodi per aprire la finestra Progetto.**

   Premete CTRL+R; selezionate Visualizza-Gestione progetti; fate clic sull'icona Gestione progetti nella barra degli strumenti.

2. **Nella finestra Progetto fate clic sul nome della form che contiene l'oggetto di cui volete esaminare le proprietà e poi fate clic sull'icona Visualizza oggetto.**

3. **Fate clic sulla freccia verso il basso della casella di riepilogo Oggetto nella finestra Proprietà e selezionate l'oggetto di cui volete esaminare le proprietà.**

4. **Fate doppio clic sul nome della proprietà che volete modificare.**

Se fate clic su un oggetto visualizzato su una form, la finestra Proprietà visualizza automaticamente le proprietà di quell'oggetto.

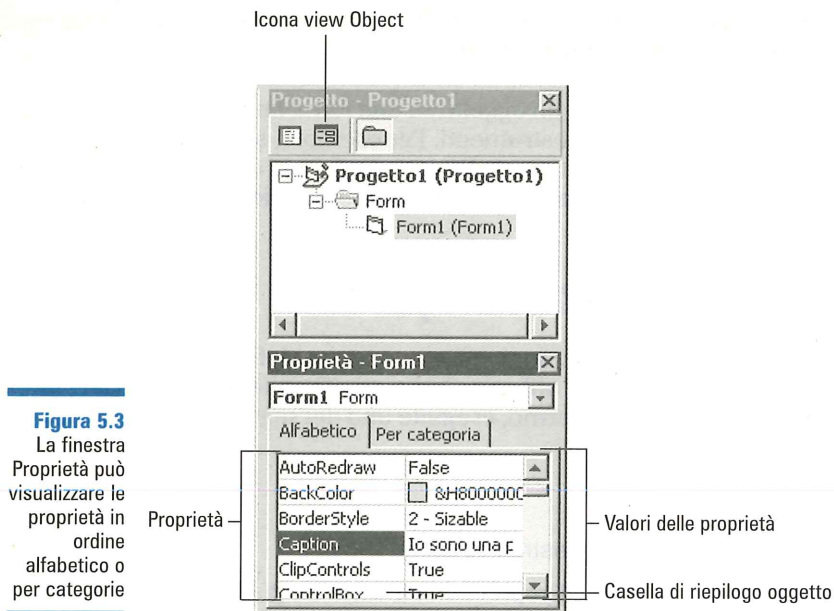



Figura 5.3 La finestra Proprietà può visualizzare le proprietà in ordine alfabetico o per categorie

Assegnare un nome agli oggetti

Ogni oggetto disegnato possiede la proprietà (Name), utilizzata da Visual Basic per identificare l'oggetto (proprio per la stessa ragione per la quale i vostri genitori vi hanno dato un nome, affinché la gente non fosse costretta a dire: "Ehi, tu" per attirare la vostra attenzione). Se fate clic sulla scheda Alfabetico nella finestra Proprietà, il nome dell'oggetto appare nella parte superiore dell'elenco.

Ogni oggetto Visual Basic deve avere un nome univoco. Se cercate di attribuire lo stesso nome a due oggetti diversi, Visual Basic si lamenterà e si rifiuterà di commettere un errore così madornale.

Quando create un oggetto Visual Basic gli attribuisce automaticamente un nome banale e generico. Per esempio, la prima volta che creerete un pulsante di comando, Visual Basic gli assegnerà il nome Command1; la seconda volta che creerete un pulsante di comando, Visual Basic lo chiamerà Command2 e così via.

Il nome dell'oggetto non compare mai sullo schermo. I nomi possono comprendere un massimo di 40 caratteri e non devono includere simboli di punteggiatura o spazi. È possibile attribuire agli oggetti qualsiasi nome, ma sarebbe meglio utilizzare il prefisso di tre lettere Visual Basic, come illustrato nella Tabella 5.2. Se tutti i programmatori utilizzassero questi prefissi, le modifiche apportate al programma Visual Basic da un altro programmatore sarebbero più semplici da leggere.

Tabella 5.2 Prefissi suggeriti per i nomi degli oggetti

Oggetto	*Prefisso suggerito*	*Nome d'esempio*
Casella di controllo (check box)	chk	chkSceltaCarriera
Casella combinata (combo box)	cbo	cboCriminiCommessi
Pulsante di comando (command button)	cmd	cmdApritiSesamo
Dati (data)	dat	datInformazioniTopSecret
Casella di riepilogo Directory (directory list box)	dir	dirAlbero
Casella di riepilogo Drive (drive list box)	drv	drvDiscoFisso
Casella di riepilogo File (file list box)	fil	filDocumenti
Form	frm	frm740Tasse
Cornice (frame)	fra	fraPulsantiRaggruppati
Barra di scorrimento orizzontale (horizontal scroll bar)	hsb	hsbTemperatura
Immagine (image)	img	imgBelDisegno
Etichetta (label)	lbl	lblNomeFalso
Riga (line)	lin	linBordo
Casella di riepilogo (list box)	lst	lstCandidati
Menu	mnu	mnuProsciuttoeUova
Riquadro immagine (picture box)	pic	picBelleImmagini
Pulsante di opzione (radio button)	opt	optStazione101
Forma (shape)	shp	shpPrendereOLasciare
Casella di testo (text box)	txt	txtAvvertimento
Barra di scorrimento verticale (vertical scroll bar)	vsb	vsbFondiRaccolti

Per modificare il nome di un oggetto utilizzate la procedura di seguito riportata.

1. **Fate clic in un punto qualunque dell'oggetto al quale volete assegnare un nome.**

   Attorno all'oggetto appariranno delle maniglie nere (per assegnare un nome a una form fate clic in un punto della form in cui non siano presenti oggetti).

2. **Aprite la finestra Proprietà.**

   Per aprirla, premete F4, selezionate Visualizza-Finestra Proprietà oppure fate clic sull'icona Finestra Proprietà nella barra degli strumenti.

3. **Fate doppio clic su (Name) e digitate il nuovo nome.**

Create titoli per gli oggetti

Oltre a un nome, la maggior parte degli oggetti (ma non tutti) possiede anche un titolo; il titolo di un oggetto è ciò che viene visualizzato sullo schermo sotto forma di testo. Alcuni titoli sono illustrati nella Figura 5.4.

Per default, il titolo e il nome di un oggetto sono identici fino a quando non li modificate. Di conseguenza, quando disegnate una casella di controllo su una form, il titolo utilizzato sarà una parola generica come Check1 e anche il nome sarà Check1.

Il titolo della form viene visualizzato nella barra del titolo della form in questione. Il titolo di un oggetto (come un pulsante di comando, un'etichetta, una casella di controllo o di testo) appare direttamente sull'oggetto, mentre il titolo di una casella di controllo o di un pulsante di opzione di solito appaiono sulla destra dell'oggetto.

I titoli hanno lo scopo di aiutare l'utente a comprendere come utilizzare il vostro programma. Un titolo può essere vuoto oppure comprendere un massimo di 255

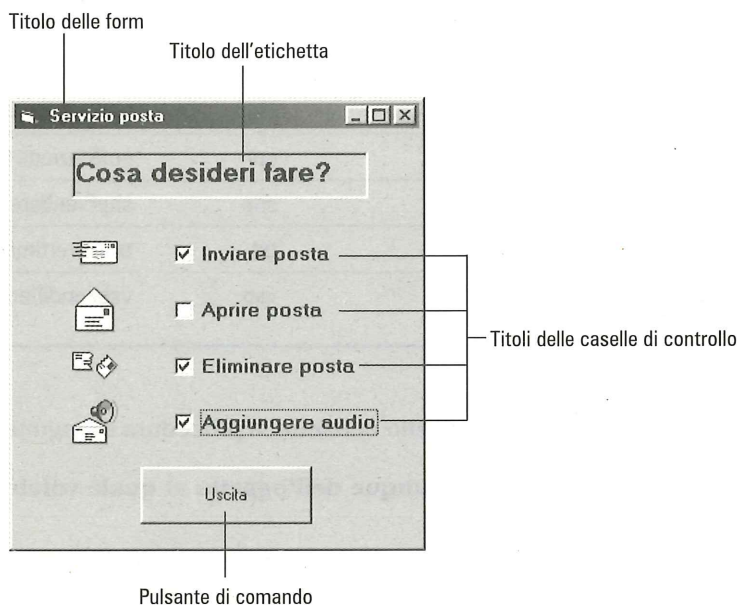



Figura 5.4 I titoli per i diversi oggetti

caratteri, compresi gli spazi, i simboli di punteggiatura e le parolacce. Ecco alcuni titoli validi.

- ✓ Ciao
- ✓ Ciao, tonto!
- ✓ So davvero che cosa sto facendo?

Per modificare il titolo di un oggetto, utilizzate la procedura di seguito riportata.

1. **Fate clic sull'oggetto di cui desiderate modificare il titolo in modo che attorno all'oggetto appaiano delle maniglie nere.**

   Per selezionare una form, fate clic in un punto qualunque della form, ma non fate clic sugli oggetti presenti sulla form.

2. **Aprite la finestra Proprietà (premendo F4).**
3. **Fate doppio clic sulla proprietà Caption e digitate un nuovo titolo.**

   Notate che Visual Basic visualizza il titolo sullo schermo nel momento stesso in cui lo digitate.

Aggiungere tasti di scelta rapida a un titolo

Oltre ad avere un bell'aspetto e a presentare le informazioni all'utente, i titoli possono essere utilizzati per creare tasti di scelta rapida (hot key) in modo che l'utente possa selezionare un oggetto senza dover fare clic su di esso con il mouse.

Per aggiungere un tasto di scelta rapida a un titolo dovete inserire la "e" commerciale (&) nel titolo dell'oggetto. A questo punto potreste chiedervi: "Perché diavolo dovrei inserire un simbolo brutto come quello?". La risposta è che questo metodo fornisce agli utenti un altro sistema per selezionare un oggetto sulla vostra interfaccia utente.

Per esempio, per premere un pulsante di comando gli utenti possono utilizzare uno dei seguenti due metodi.

- ✓ Fare clic con il mouse sul pulsante di comando.
- ✓ Premere TAB fino a evidenziare il pulsante e poi premere la BARRA SPAZIATRICE oppure INVIO (ovvio, non è vero?).

Se nel titolo del pulsante avete utilizzato il carattere &, l'utente potrà premere un pulsante schiacciando ALT più la lettera che segue la &. Per esempio, se un pulsante di comando ha il titolo &Uscita, il titolo viene visualizzato con la U sottolineata; per selezionare questo pulsante è sufficiente premere ALT+U. Se invece il pulsante di comando ha il titolo Uscita, nel titolo è la s ad apparire sottolineata e l'utente può premere ALT+S per selezionarlo (Figura 5.5).

Mettete alla prova le vostre nuove conoscenze

1. **Perché è importante conoscere il tipo di utenti che potrebbero utilizzare il vostro programma?**

   a. Per poter creare false aspettative e confondere i vostri utenti.

   b. Per personalizzare l'interfaccia utente in base alle loro aspettative ed esperienze.

   c. Perché qualcuno potrebbe avere degli amici nelle alte sfere.

   d. Sapere chi sono gli utenti non è importante finché pagano il programma in contanti.

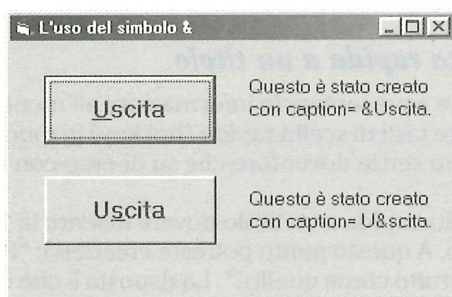
2. **Un'interfaccia utente deve essere come una mappa. Spiegate questa frase.**

   a. Perché dovrei spiegarla? Sei stato tu a scriverla.

   b. Il Touring Club sta scrivendo del software.

   c. Questa frase è una similitudine, ovvero un modo creativo di esprimere delle analogie.

   d. L'interfaccia utente deve sempre indicare agli utenti in che punto del programma si trovano.

Figura 5.5 I due pulsanti di comando che illustrano l'utilizzo della "e" commerciale (&)

Modificare il tipo di carattere del titolo di un oggetto

Quando create un titolo Visual Basic lo visualizza sullo schermo con caratteri normali. I tipi creativi che desiderano dare ai titoli un'aria più raffinata, possono modificare il tipo di carattere, lo stile e le dimensioni del titolo per renderlo più vivace.

I *tipi di carattere* permettono di modificare la forma dei caratteri del testo da visualizzare; di solito Visual Basic utilizza il tipo di carattere MS Sans Serif, ma potete utilizzare qualsiasi tipo di carattere installato sul vostro computer. L'MS Sans Serif è simile all'Helvetica, mentre l'MS Serif di Visual Basic è più simile al Times Roman.

Per modificare il tipo di carattere di un titolo utilizzate la procedura di seguito riportata.

1. **Fate clic sull'oggetto di cui desiderate modificare il titolo.**

2. **Aprite la finestra Proprietà (selezionando Visualizza-Finestra Proprietà).**
3. **Fate doppio clic sulla proprietà Tipo di carattere.**

   Visual Basic visualizza una finestra di dialogo che contiene tutti i tipi di carattere che potete utilizzare, come illustrato nella Figura 5.6.
4. **Fate clic sul tipo di carattere desiderato e poi su OK.**

   Visual Basic modifica immediatamente il tipo di carattere del titolo.

I tipi di carattere vi permettono di essere creativi, ma possono anche disorientare l'utente, specialmente se scegliete caratteri troppo bizzarri e del tutto diversi da quanto normalmente è presente in natura. Per evitare confusione, permettete a Visual Basic di utilizzare il tipo di carattere di default MS Sans Serif, a meno che non vi siano ragioni specifiche per modificarlo.

Se scegliete un tipo di carattere decisamente stravagante, quel carattere potrebbe non essere presente nei computer degli altri utenti. Quando si utilizzano i tipi di carattere, sarebbe meglio utilizzare quelli più comuni che si trovano su tutti i computer; in caso contrario, i titoli del vostro programma potrebbero avere un aspetto veramente strano.

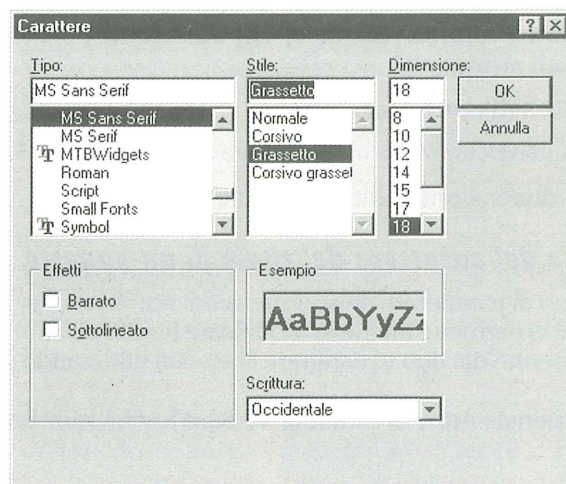



Figura 5.6 La finestra di dialogo carattere

Modificare le dimensioni del titolo di un oggetto

È possibile modificare le dimensioni del titolo rimpicciolendolo o ingrandendolo in modo da adattarlo all'oggetto. A seconda del tipo di carattere selezionato, Visual Basic offre una serie di dimensioni possibili.

Per esempio, se selezionate il tipo di carattere MS Sans Serif, Visual Basic offre le seguenti scelte:

- 8
- 10
- 12
- 14
- 18
- 24

Ovviamente, maggiori sono le dimensioni dei tipi di carattere, più strani appariranno i vostri titoli. Il metodo migliore è utilizzare un'unica dimensione per evitare di confondere le idee all'utente.

Per definire le dimensioni del tipo di carattere dei vostri titoli utilizzate la procedura di seguito riportata.

1. **Fate clic sull'oggetto di cui desiderate modificare le dimensioni.**
2. **Aprite la finestra Proprietà (facendo clic sull'icona Finestra Proprietà nella barra degli strumenti).**
3. **Fate doppio clic sulla proprietà Font.**

   Visual Basic visualizza una finestra di dialogo analoga (Figura 5.6).
4. **Selezionate le dimensioni desiderate e fate clic su OK.**

Modificare lo stile del carattere del titolo di un oggetto

Se la modifica del tipo di carattere e delle dimensioni non basta per rallegrarvi la giornata, Visual Basic vi permette anche di modificare lo stile dei titoli. La quantità di stili disponibile dipende dal tipo di carattere che state utilizzando per il titolo.

Per esempio, se selezionate il tipo di carattere MS Sans Serif, Visual Basic vi offre le seguenti scelte.

- Normale
- *Corsivo*
- **Grassetto**
- ***Grassetto corsivo***
- Sottolineato
- ~~Barrato~~

Questi stili diversi sono illustrati nella Figura 5.7. Volendo, potete persino combinare due o più stili per conferire una maggiore enfasi.

Per impostare uno o più di questi stili, utilizzate la procedura di seguito riportata.

1. **Fate clic sull'oggetto di cui desiderate modificare il titolo.**
2. **Aprite la finestra Proprietà, premendo F4.**
3. **Fate doppio clic sulla proprietà Font.**

   Visual Basic visualizza una finestra di dialogo analoga a quella illustrata nella Figura 5.6.
4. **Fate clic sullo stile che desiderate utilizzare e fate clic su OK.**

   Visual Basic modifica immediatamente l'aspetto del titolo.

Modificare i colori di sfondo e di primo piano dei titoli

I titoli vengono solitamente visualizzati in nero, bianco e nei toni di grigio. Per rendere più variopinti e meno noiosi i vostri titoli, potete modificare i colori di sfondo e di primo piano. La proprietà BackColor di un oggetto rappresenta il colore di sfondo, mentre la proprietà ForeColor rappresenta il colore di primo piano, come illustrato nella Figura 5.8.

A differenza di altri tipi di oggetti, i pulsanti di comando hanno soltanto la proprietà BackColor, che si limita a modificare il colore che circonda il titolo quando il pulsante di comando viene evidenziato.

Per modificare il colore attorno al titolo di un oggetto utilizzate la procedura di seguito riportata.

1. **Fate clic sull'oggetto di cui desiderate modificare il colore di sfondo (o di primo piano).**

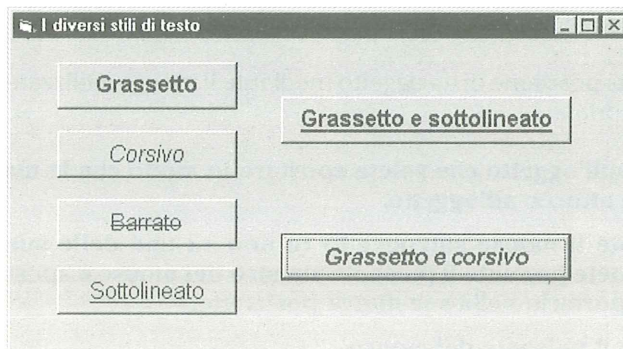
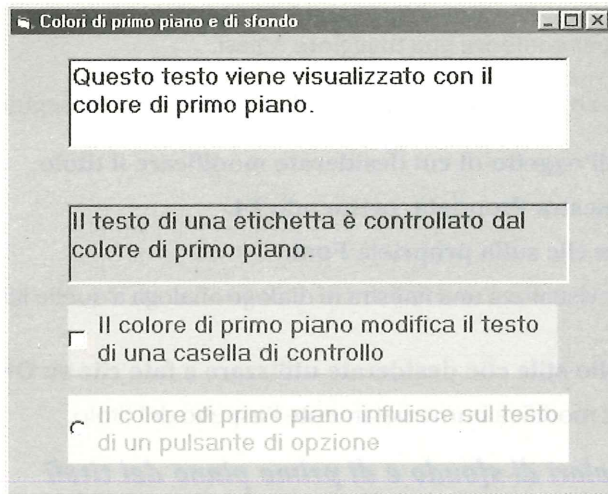



Figura 5.7 Pulsanti di comando che illustrano i diversi stili del testo

Figura 5.8 I caratteri vengono visualizzati con i colori di primo piano; tutto il resto appare nel colore di sfondo

2. **Aprite la finestra Proprietà, premendo F4.**
3. **Fate doppio clic sulla proprietà BackColor (o ForeColor) e poi fate clic sulla scheda Tavolozza.**

   Visual Basic visualizza una tavolozza di colori.
4. **Fate clic sul colore desiderato.**

   Visual Basic obbedisce all'istante.

Spostare gli oggetti sullo schermo

Gli oggetti possono apparire in un punto qualsiasi su una form. Visual Basic offre due metodi per definire la posizione di un oggetto su una form.

- ✓ Mediante il mouse.
- ✓ Modificando le proprietà Left e Top nella finestra Proprietà.

Per modificare la posizione di un oggetto mediante il mouse, utilizzate la procedura di seguito riportata.

1. **Fate clic sull'oggetto che volete spostare in modo che le maniglie nere appaiano attorno all'oggetto.**
2. **Posizionate il mouse sull'oggetto (e non su una delle sue maniglie), quindi tenete premuto il pulsante sinistro del mouse e spostate l'oggetto fino a portarlo nella sua nuova posizione.**
3. **Rilasciate il pulsante del mouse.**

Nel caso non abbiate ancora capito che cosa sta succedendo, posizionando un oggetto abbiamo comunicato a Visual Basic che da ora in poi quel determinato oggetto dovrà essere visualizzato in quella determinata posizione. Utilizzate il mouse ogni volta che desiderate spostare un oggetto rapidamente, senza preoccuparvi di posizionarlo esattamente sullo schermo.

Per spostare un oggetto e posizionarlo in modo più corretto utilizzate la finestra Proprietà e inserite i valori per le proprietà Left e Top.

Nel caso degli oggetti, la proprietà Left misura la distanza dal bordo sinistro della form al bordo sinistro dell'oggetto, mentre la proprietà Top misura la distanza dal margine superiore della form al margine superiore dell'oggetto.

Nel caso delle form, la proprietà Left misura la distanza dal bordo sinistro dello schermo al bordo sinistro della form, mentre la proprietà Top misura la distanza dalla cima dello schermo alla cima della form.

Per modificare la posizione di un oggetto utilizzando la finestra Proprietà, seguite la procedura di seguito riportata.

1. **Fate clic sull'oggetto che volete spostare per far apparire le maniglie nere.**

   Se volete modificare la posizione di una form, fate clic in un punto qualunque della form per far apparire le maniglie attorno ai bordi della form.

2. **Aprite la finestra Proprietà, premendo F4.**
3. **Fate doppio clic sulla proprietà Left e digitate un nuovo valore.**
4. **Fate doppio clic sulla proprietà Top e digitate un nuovo valore.**

Cancellare gli oggetti dalla faccia della terra

Talvolta potrebbe capitarvi di disegnare un oggetto inutile e di doverlo alla fine cancellare.

Per cancellare un oggetto utilizzate la procedura di seguito riportata.

1. **Fate clic sull'oggetto che volete cancellare.**
2. **Premete CANC oppure selezionate Modifica-Elimina.**

Se premete CTRL+Z subito dopo avere cancellato un oggetto, potete annullare l'operazione di cancellazione dell'oggetto appena effettuata.

Copiare gli oggetti perché siete troppo stanchi per disegnarne di nuovi

Dopo avere disegnato un oggetto delle dimensioni desiderate, potreste decidere di copiarlo, invece di crearne uno nuovo ed essere così costretti a modificarne le dimensioni. Per copiare un oggetto, utilizzate la procedura di seguito riportata.

1. **Fate clic sull'oggetto che volete copiare.**
2. **Premete CTRL+C oppure fate clic sull'icona Copy.**
3. **Premete CTRL+V oppure fate clic sull'icona Paste.**

   Visual Basic visualizza una finestra di dialogo in cui vi viene chiesto se volete creare una matrice di controlli. Se sapete che cos'è una matrice di controlli e volete crearne una, fate clic su Sì; in caso contrario, fate clic su No. Visual Basic visualizza una copia del vostro oggetto nell'angolo superiore sinistro della form.

   Un *array di controllo* consente di creare due o più oggetti che condividono lo stesso nome. In questo modo, due o più oggetti possono condividere la medesima procedura di evento. Se tutto questo sembra non avere alcun senso per voi, fate finta che questo paragrafo non esista.

4. **Spostate questa copia dell'oggetto nel punto dello schermo desiderato.**

Selezionare più di un oggetto da spostare, copiare o cancellare

Prima di poter spostare, copiare o cancellare un oggetto, dovete selezionarlo facendo clic su di esso. Se invece volete spostare, copiare o cancellare più di un oggetto contemporaneamente, avete due possibilità.

- ✓ Utilizzare il mouse per selezionare più oggetti.
- ✓ Fare clic su più oggetti tenendo premuto CTRL oppure MAIUSC.

Per selezionare più oggetti con il mouse utilizzate la procedura di seguito riportata.

1. **Posizionate il mouse nell'angolo superiore sinistro del gruppo di oggetti che volete selezionare (ma non direttamente su alcuno degli oggetti da selezionare).**
2. **Tenete premuto il pulsante sinistro del mouse mentre trascinate il mouse nell'angolo inferiore sinistro del gruppo di oggetti che volete selezionare (Figura 5.9).**

   Visual Basic visualizza una riga tratteggiata attorno agli oggetti selezionati.

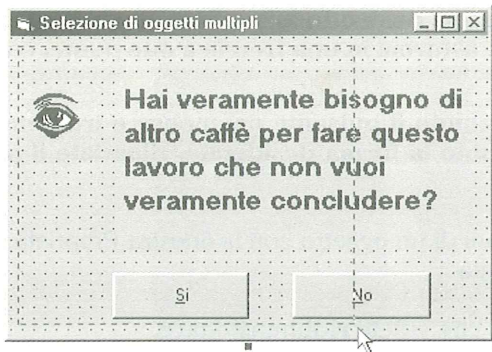



Figura 5.9 La selezione di oggetti multipli con il mouse

3. **Rilasciate il pulsante del mouse.**

   Visual Basic visualizza un rettangolo grigio attorno agli oggetti selezionati.

Per fare clic su più oggetti tenendo premuto CTRL o MAIUSC, utilizzate la procedura di seguito riportata.

1. **Fate clic sul primo oggetto che volete selezionare.**

   Visual Basic visualizza le maniglie nere attorno all'oggetto.

2. **Puntate al secondo oggetto che volete selezionare.**
3. **Premete e tenete premuto CTRL o MAIUSC prima di fare clic sul secondo oggetto.**

   Visual Basic visualizza dei rettangoli grigi attorno a questo oggetto e a ciascuno degli oggetti selezionati in precedenza.

4. **Ripetete i passi 2 e 3 finché avrete selezionato tutti gli oggetti desiderati.**

Modificare le dimensioni degli oggetti

La fase successiva alla creazione di un oggetto consiste nel definirne le dimensioni. Visual Basic offre due metodi per modificare le dimensioni di un oggetto.

- ✓ Utilizzando il mouse.
- ✓ Modificando le proprietà Height e Width nella finestra Proprietà.

Per modificare le dimensioni di un oggetto con il mouse, utilizzate la procedura di seguito riportata.

1. **Fate clic sull'oggetto che volete ridimensionare.**

   Attorno ai bordi dell'oggetto appaiono delle piccole maniglie nere.

2. **Spostate il mouse su una delle otto maniglie dell'oggetto finché il puntatore del mouse si trasforma in una freccia a due punte, come illustrato nella Figura 5.10.**
3. **Premete e tenete premuto il pulsante del mouse e trascinate; quando l'oggetto avrà raggiunto la forma desiderata, rilasciate il pulsante del mouse.**

Per modificare le dimensioni di un oggetto con la finestra Proprietà, utilizzate la procedura di seguito riportata.

1. **Fate clic sull'oggetto che volete ridimensionare.**
2. **Aprite la finestra Proprietà, premendo F4.**
3. **Fate doppio clic sulla proprietà Height, digitate un nuovo valore e poi premete INVIO.**
4. **Fate doppio clic sulla proprietà Width, digitate un nuovo valore e poi premete INVIO.**

Utilizzate il metodo del mouse quando non è fondamentale ottenere dimensioni precise. Modificate manualmente le proprietà Height e Width quando volete la massima precisione oppure quando vi sentite particolarmente pignoli.

Figura 5.10 Ridimensionare un oggetto

Definire la proprietà TabIndex dei vostri oggetti

La proprietà TabIndex determina la sequenza in base alla quale Visual Basic evidenzia i pulsanti quando l'utente preme uno dei seguenti tasti: TAB, ↑, ↓, →, ← o MAIUSC+TAB.

- ✓ I tasti TAB, ↓ e → evidenziano l'oggetto utilizzando il valore di TabIndex immediatamente superiore.
- ✓ I tasti MAIUSC+TAB, ↑ e ← evidenziano l'oggetto utilizzando il valore di TabIndex immediatamente inferiore. Per evidenziare i pulsanti di opzione potete utilizzare soltanto i tasti di direzione, ma non i tasti TAB o MAIUSC+TAB.
- ✓ La BARRA SPAZIATRICE o INVIO selezionano un oggetto evidenziato.

Alcuni oggetti, come i riquadri immagine e gli oggetti dei menu, non hanno la proprietà TabIndex e quindi non possono essere evidenziati premendo un tasto.

Un oggetto la cui proprietà TabIndex vale 0 appare come evidenziato non appena il programma viene eseguito. Se l'utente preme TAB, l'oggetto la cui proprietà TabIndex è pari a 1 sarà evidenziato e così via. Il primo oggetto creato ha una proprietà TabIndex pari a 0, il secondo ha una proprietà TabIndex pari a 1 e così via.

L'unico metodo per evidenziare un oggetto memorizzato all'interno di una cornice è premere TAB oppure fare clic sull'oggetto con il mouse. Quando un oggetto all'interno di una cornice viene evidenziato, premendo i tasti di direzione vengono selezionati soltanto gli altri pulsanti di comando o oggetti di quella cornice. Sebbene la maggior parte degli utenti utilizzerà il mouse per evidenziare e selezionare gli oggetti, alcuni potrebbero anche non utilizzare questo metodo. In questi casi rari, la tastiera è l'unico sistema concesso a questi utenti antiquati per selezionare gli oggetti dell'interfaccia utente.

Per modificare la proprietà TabIndex di un oggetto, utilizzate la procedura di seguito riportata.

1. **Fate clic sull'oggetto che volete modificare.**
2. **Aprite la finestra Proprietà, premendo F4.**
3. **Fate doppio clic sulla proprietà TabIndex e digitate un numero (come 1 o 4).**

Ogni volta che modificate la proprietà TabIndex di un pulsante, Visual Basic rinumera automaticamente i valori TabIndex degli altri pulsanti. In questo modo è impossibile che due pulsanti abbiano valori TabIndex identici.

Se avete creato molti oggetti potete impostare la proprietà TabIndex in modo rapido e semplice utilizzando la procedura di seguito riportata.

1. **Fate clic sull'oggetto che volete evidenziare per ultimo (l'oggetto che avrà il valore TabIndex massimo).**
2. **Aprite la finestra Proprietà, premendo F4.**
3. **Fate clic sulla proprietà TabIndex e digitate 0.**

4. **Fate clic sull'oggetto che volete evidenziare come penultimo.**
5. **Ripetete i passi da 2 a 4 fino a quando avrete impostato a 0 la proprietà TabIndex di tutti gli oggetti.**

Se utilizzate questo metodo, l'ultimo oggetto selezionato avrà un valore TabIndex pari a 0, il penultimo un valore pari a 1 e così via.

Se per qualche ragione, non volete che l'utente possa evidenziare un oggetto premendo TAB, impostate la proprietà TabStop di questo oggetto su False.

Disattivare gli oggetti

Se non volete che l'utente utilizzi un particolare oggetto (per esempio un pulsante di comando, una casella di controllo o un riquadro di immagine) potete disattivarlo; questo apparirà come illustrato nella Figura 5.11. Un oggetto disattivato dice all'utente: "Talvolta potete fare clic su questo oggetto, ma non adesso".

Per disattivare un oggetto, utilizzate la procedura di seguito riportata.

1. **Fate clic sull'oggetto che volete disattivare.**
2. **Aprite la finestra Proprietà, premendo F4.**
3. **Fate clic sulla proprietà Enabled e impostatela a False.**

Dato che un oggetto disattivato non fa nulla, alla fine dovrete modificare il valore di questa proprietà durante la fase di esecuzione utilizzando un comando BASIC.

Per offrirvi un assaggio dell'incredibile potenza del codice BASIC, ecco come il BASIC attiva e disattiva un pulsante. Per attivare un pulsante impostate la proprietà Enabled a True. Il seguente esempio attiva un pulsante denominato cmdUscita.

```
cmdUscita.Enabled = True
```

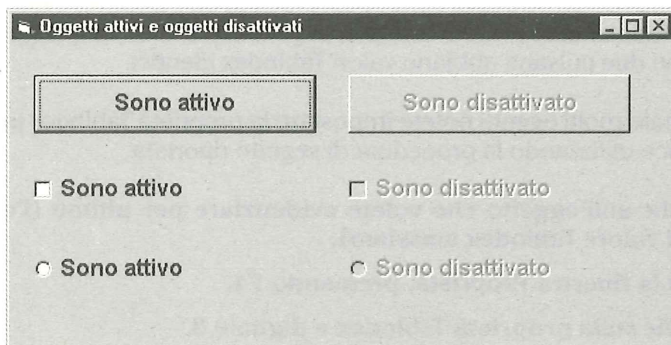



Figura 5.11 Oggetti attivi e oggetti disattivati

Per disattivare un pulsante utilizzando il codice BASIC, impostate la proprietà Enabled a False. Il seguente esempio disattiva un pulsante denominato cmdUscita.

```
cmdUscita.Enabled = False
```

È possibile attivare e disattivare i pulsanti utilizzando del codice BASIC mentre il programma è in esecuzione. In questo modo potrete attivare e disattivare i pulsanti in risposta alle azioni compiute dall'utente (che digita, sposta il mouse, batte freneticamente sulla tastiera e così via).

Rendere invisibili gli oggetti

Invece di disattivare un oggetto (metodo che si prende gioco dell'utente, dato che l'oggetto c'è, ma non è disponibile) potete farlo scomparire completamente.

Per far scomparire un oggetto utilizzate la procedura di seguito riportata.

1. **Fate clic sull'oggetto che volete far scomparire.**
2. **Premete F4 oppure selezionate Finestra Proprietà dal menu Visualizza per aprire la finestra Proprietà.**
3. **Fate clic sulla proprietà Visible e impostatela a False.**

Potete far scomparire degli oggetti anche utilizzando il codice BASIC. A questo scopo impostate la proprietà Visible dell'oggetto a False. Il seguente esempio fa scomparire un pulsante di comando denominato cmdNuovo:

```
cmdNuovo.Visible = False
```

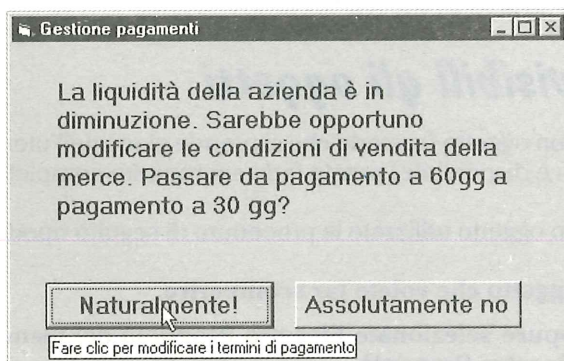
Proprio come avviene per gli oggetti disattivati, gli oggetti invisibili non sono utilizzabili a meno che, di tanto in tanto, li facciate ricomparire sullo schermo. A questo scopo dovete utilizzare il codice BASIC per impostare la proprietà Visible a True. Il seguente esempio rende visibile il pulsante di comando denominato cmdNuovo:

```
cmdNuovo.Visible = True
```

Utilizzare le descrizioni

Malgrado l'interfaccia utente standard di Windows 95/98/NT molti utenti potrebbero non capire quali operazioni svolgono effettivamente i diversi oggetti del vostro programma. Molti utenti, invece di fare clic a casaccio su questi oggetti indefiniti (rischiando così di distruggere i loro dati), rimangono paralizzati dalla paura e alla fine non utilizzeranno mai alcune parti del programma.

Per superare questo ostacolo tutti i vostri oggetti possono visualizzare delle descrizioni (tooltip). Una descrizione è un breve testo che illustra la funzione svolta da un oggetto particolare. La descrizione rimane invisibile fino a quando l'utente sposta il cursore del mouse sull'oggetto lasciandolo lì per alcuni secondi. Poi il programma Visual Basic ritorna in vita e visualizza il testo della descrizione, come illustrato nella Figura 5.12.

Figura 5.12 La visualizzazione della descrizione di un oggetto

Per creare il testo della descrizione per i vostri oggetti utilizzate la procedura di seguito riportata.

1. **Fate clic su un oggetto della form.**
2. **Premete F4 per aprire la finestra Proprietà.**
3. **Fate doppio clic sulla proprietà ToolTipText.**
4. **Digitate una breve spiegazione che volete venga visualizzata ogniqualvolta l'utente lascia il cursore del mouse sopra l'oggetto.**

Provate anche voi a modificare i titoli

Il seguente programma d'esempio vi permette di modificare il titolo della form digitando il nuovo titolo in una casella di testo e facendo clic sul pulsante di comando cmdTitolo. Create un'interfaccia utente in base alla Tabella 5.3 e sperimentate da soli i sorprendenti poteri di Visual Basic.

Tabella 5.3 Proprietà da modificare per CAPTION.VBP

Oggetto	*Proprietà*	*Valore*
Form	Caption	L'incredibile titolo trasformista
	Height	3600
	Width	4800
Label1	(Name)	lblTestata
	Caption	Questo titolo può essere modificato facendo clic sul pulsante Modifica
	Height	495
	Left	840
	Top	240
	Width	2895
Label2	Caption	Digitare un nuovo titolo qui sotto
	Height	255
	Left	1200
	Top	1200
	Width	2415
Text1	(Name)	txtTitolo
	Altezza	495
	Left	8410
	Multiline	True
	Text	(vuoto)
	Top	1560
	Width	3015
Command1	(Name)	cmdTitolo
	Caption	Modifica
	Height	495
	Left	360
	TodTiptext	Fare clic qui per modificare il titolo

(continua)

Tabella 5.3

Oggetto	*Proprietà*	*Valore*
Top	2400	
	Width	1695

Coloro che odiano scrivere possono esaminare il programma CAPTION.VBP incluso sul CD-ROM.

```
Private Sub cmdTitolo_Click()
  lblTestata.Caption = txtTitolo.Text
End Sub
```

Quando eseguite questo programma digitate un nuovo titolo nella casella di testo e fate clic sul pulsante di comando definito Modifica titolo. Visual Basic visualizza immediatamente il testo appena digitato nell'etichetta in alto.

Capitolo 6

Form e pulsanti

In questo capitolo

- Creare form e disegnare bordi
- Determinare l'ordine di visualizzazione delle form
- Creare i pulsanti

La parte principale di un'interfaccia utente è costituita da una finestra, che Visual Basic definisce form. Un programma Visual Basic deve avere perlomeno una form, sebbene la maggior parte dei programmi ne utilizzi due o anche di più.

Un tipico programma potrebbe, per esempio, utilizzare una form per visualizzare un elenco di pulsanti di comando da selezionare facendo clic. Quando l'utente fa clic su un pulsante di comando, appare una seconda form che visualizza informazioni quali i nomi, gli indirizzi e i numeri telefonici delle persone che vi devono dei soldi.

Il CD-ROM in fondo al presente libro contiene un semplice programma in Visual Basic che mostra come modificare lo sfondo di una form. Sentitevi liberi di fare pratica con questo programma e di modificarlo in modo da scoprire quanto è semplice cambiare l'aspetto di una form.

Creare una form

Visual Basic vi offre due metodi per creare una form.

- ✓ Creare una form vuota.
- ✓ Utilizzare un modello di form.

La creazione di una form vuota è utile quando volete progettare l'aspetto della form da soli. Un metodo più rapido per la creazione delle form è l'utilizzo di un modello di form, che fornisce form già progettate per la visualizzazione del suggerimento del giorno, di un browser Web o della schermata iniziale del vostro programma.

Anche se utilizzate un modello di form, dovete personalizzare la form in base alle esigenze del vostro programma, ma perlomeno non dovrete perdere tempo a disegnare gli oggetti sulla form partendo da zero. Nella Figura 6.1 sono illustrati i diversi modelli che Visual Basic è in grado di creare.

- **Form.** Crea una form vuota.
- **Dialog.** Crea l'impalcatura di una finestra di dialogo con un pulsante OK e un pulsante Annulla.
- **Finestra Informazioni su.** Visualizza le informazioni sul vostro programma.
- **Finestra Accesso.** Obbliga l'utente a digitare un ID utente e una password prima di poter utilizzare il programma.
- **Finestra Opzioni.** Visualizza una finestra di dialogo con alcune schede, che permette al programma di offrire le opzioni per la personalizzazione.
- **Schermata iniziale.** Visualizza il nome del programma e un logo ogniqualvolta il programma viene caricato.
- **Schermata Suggerimento del giorno.** Fornisce alcuni suggerimenti per l'utilizzo del programma.
- **Schermata Accesso ODBC.** Visualizza le opzioni per collegare il programma a un database.
- **Browser Web.** Aggiunge un browser Web al programma.
- **Creazione guidata form dati VB.** Crea automaticamente una form per visualizzare e modificare le informazioni di database.

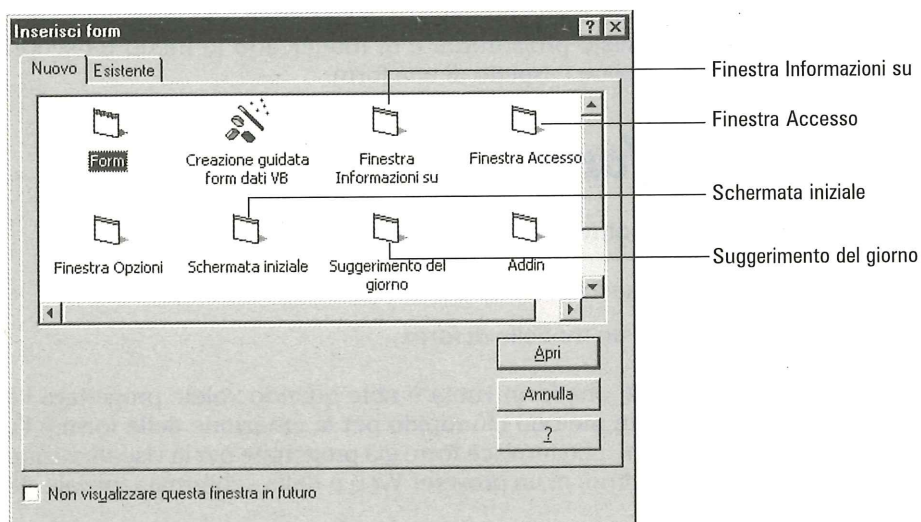



Figura 6.1 I diversi modelli che Visual Basic può creare

Per creare una form per il programma utilizzate la procedura di seguito riportata.

1. **Selezionate Progetto-Inserisci form oppure fate clic sull'icona Inserisci form sulla barra degli strumenti e fate clic su Form, come illustrato nella Figura 6.2.**

   Appare la finestra di dialogo Inserisci form.

2. **Fate clic su un'icona nella finestra di dialogo Inserisci form, come per esempio Form (che aggiunge una forma vuota), Web Broser o Splash Screen, quindi selezionate Apri.**

Salvare le form

Dopo avere creato una form, dovete salvarla (se non per motivi religiosi, almeno per ragioni pratiche), così da non doverla ricreare nuovamente in un secondo momento. Visual Basic vi offre due metodi per salvare una form.

- ✓ Selezionare File-Salva file.
- ✓ Premere CTRL+S.

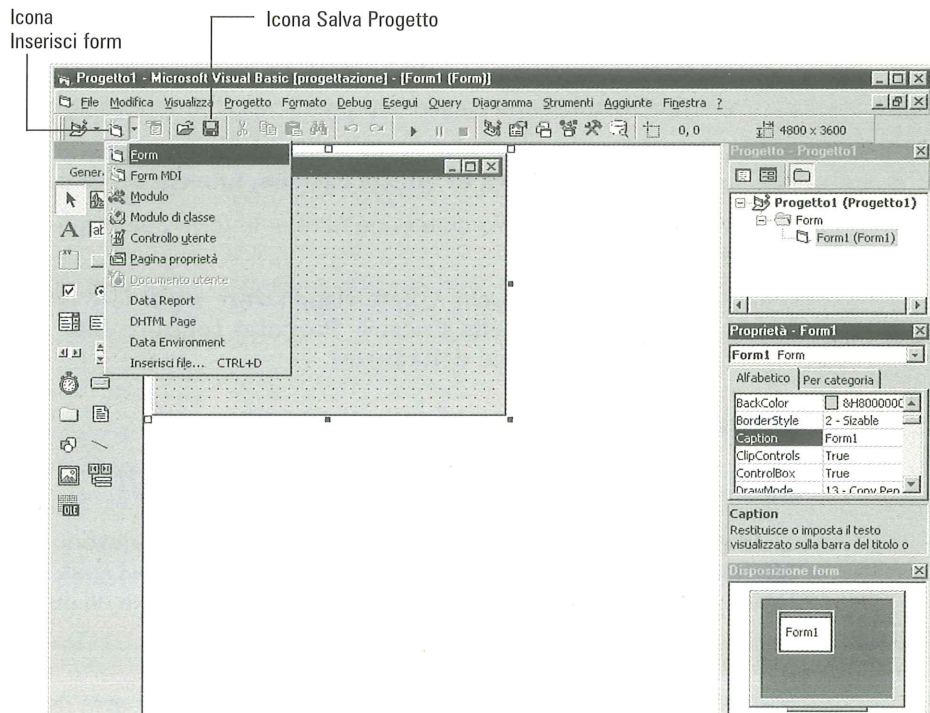



Figura 6.2 Inserire una form in un programma

Se avete due o più form visualizzate sullo schermo e volete salvare le variazioni apportate a tutte le form, selezionate File-Salva progetto oppure fate clic sull'icona Salva progetto (Figura 6.2). Questo comando salva automaticamente tutti i file (elencati nel file di progetto VBP) che compongono il vostro programma Visual Basic completo.

Prendete l'abitudine di salvare periodicamente le vostre form. Se dovesse verificarsi un malfunzionamento del computer, un calo di tensione oppure i terroristi attaccassero la vostra casa e crivellassero il vostro computer di colpi, voi perdereste soltanto le modifiche apportate dopo l'ultimo salvataggio.

Visualizzare diverse form

La maggior parte dei programmi Visual Basic necessita di due o più form. Per impedire che le diverse form creino confusione sullo schermo, Visual Basic visualizza gentilmente una form alla volta.

Quindi, se volete passare a esaminare un'altra form, utilizzate la procedura di seguito riportata.

1. **Scegliete uno dei seguenti metodi per attivare la finestra Progetto.**
   - Selezionate Visualizza-Gestione progetti.
   - Premete CTRL+R.
   - Fate clic sull'icona Gestione progetti nella barra degli strumenti.
2. **Cercate la cartella Form, come illustrato nella Figura 6.3. Se un segno più appare a sinistra di questa cartella, fate clic su di esso.**

   Visual Basic visualizza un elenco di tutte le form che costituiscono il vostro programma.
3. **Fate clic sulla form che volete visualizzare e poi fate clic sull'icona Visualizza oggetto nella finestra Progetto (oppure fate semplicemente doppio clic sul nome della form).**

Posizionare una form con la finestra Disposizione form

La finestra Disposizione form vi permette di scegliere dove devono apparire le form quando il programma è in esecuzione, come illustrato nella Figura 6.4. Se la finestra Disposizione form non è visualizzata, utilizzate uno dei seguenti due metodi.

- ✓ Selezionate Visualizza-Finestra Disposizione form.
- ✓ Fate clic sull'icona Finestra Disposizione form nella barra degli strumenti.

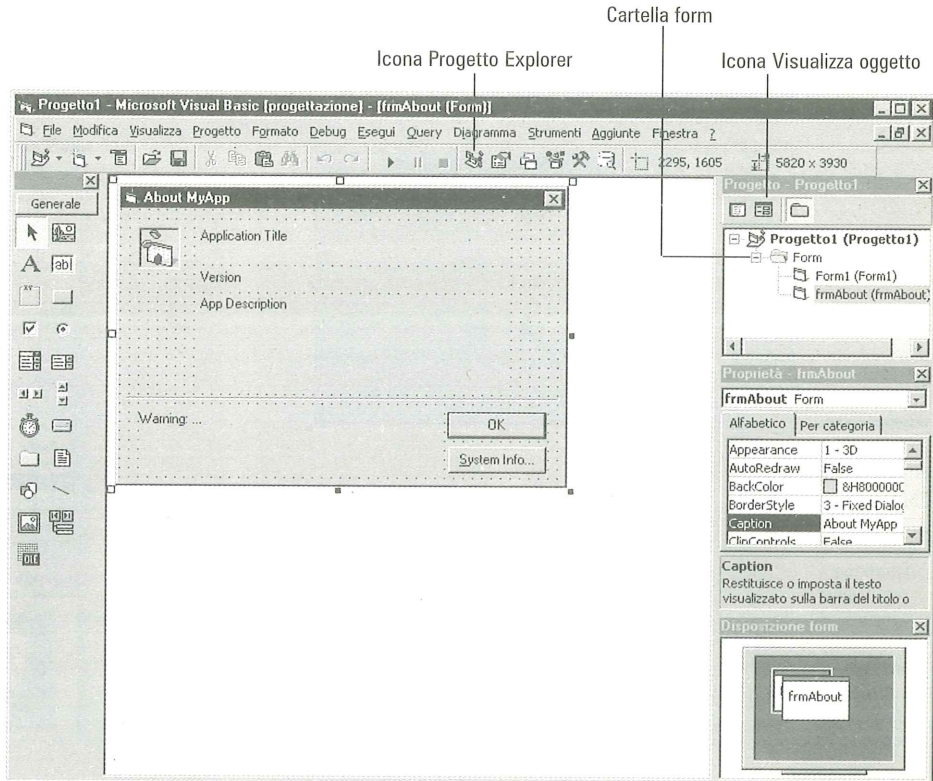



Figura 6.3 Visualizzare una form diversa

La finestra Disposizione form visualizza le form all'interno di una schermata ridotta. Per spostare la posizione di una form utilizzate la procedura di seguito riportata.

1. **Nella finestra Disposizione form, spostate il cursore sulla form che volete spostare.**

   Il cursore del mouse si trasforma in una freccia a quattro punte.

2. **Tenete premuto il pulsante sinistro del mouse e trascinate il mouse per spostare la form nella nuova posizione.**

3. **Rilasciate il pulsante sinistro del mouse.**

A prescindere dal punto in cui la form potrebbe apparire sullo schermo durante la fase di modifica, la finestra Disposizione form controlla la posizione effettiva della form quando il programma è in esecuzione.

SUGGERIMENTO

Per correggere rapidamente la posizione di una form all'interno della finestra Disposizione form, fate clic sulla form con il pulsante destro del mouse, evidenziate

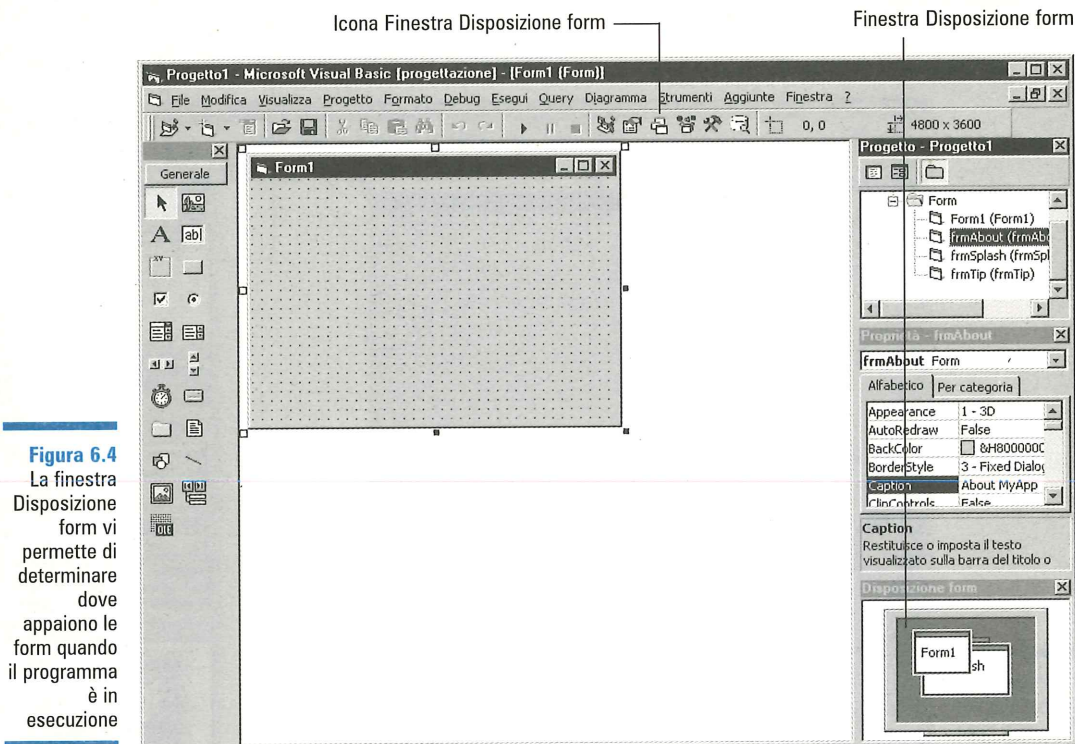



Figura 6.4 La finestra Disposizione form vi permette di determinare dove appaiono le form quando il programma è in esecuzione

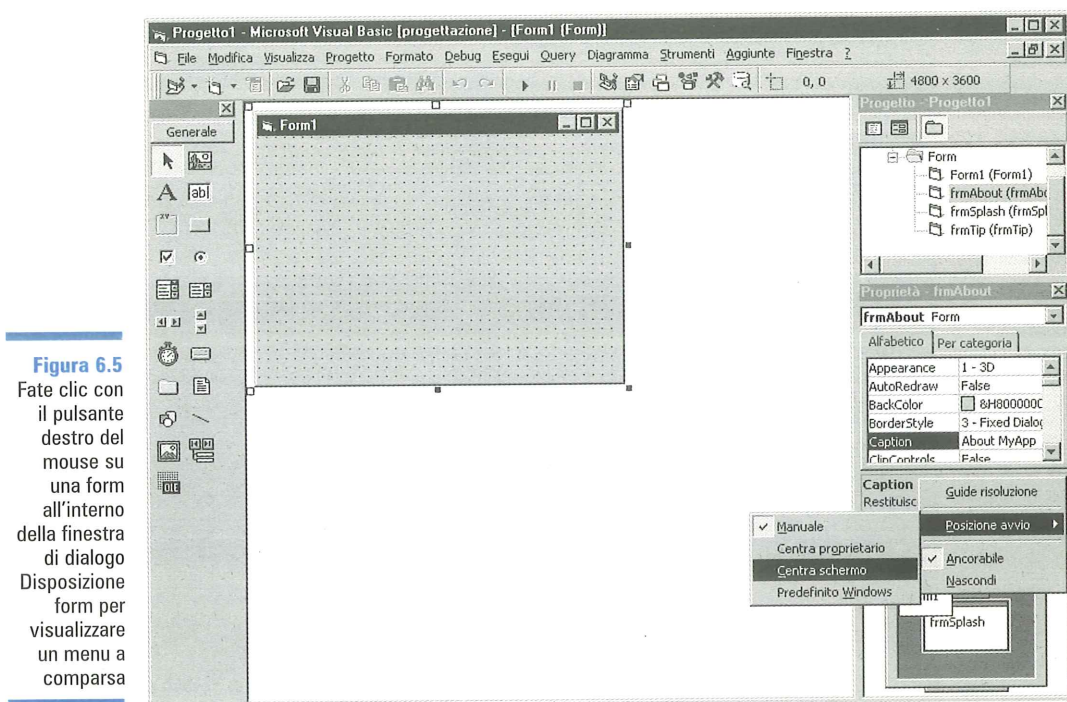
Revisione avvio, quindi selezionate un'opzione dal menu a comparsa, come illustrato nella Figura 6.5. Oppure, potete modificare la proprietà StartUpPosition direttamente all'interno della finestra Proprietà.

Disegnare i bordi attorno alle form

I bordi conferiscono un aspetto gradevole alle form. Modificando la proprietà BorderStyle è possibile attribuire alle form sei tipi diversi di bordo, come illustrato nella Figura 6.6.

Oltre a conferire un aspetto gradevole alle form, lo stile del bordo influisce anche sulla possibilità dell'utente di spostare o ridimensionare la form.

- ✓ Lo stile 0 (None) non visualizza un bordo attorno alla form, ma gli oggetti contenuti nella form vengono comunque visualizzati. Un utente non può spostare, ridimensionare o ridurre a icona questo tipo di form.
- ✓ Lo stile 1 (Fixed Single) visualizza l'icona del menu di controllo, la barra del titolo e il pulsante di chiusura. Gli utenti possono spostare ma non ridimensionare questo tipo di form.

Figura 6.5 Fate clic con il pulsante destro del mouse su una form all'interno della finestra di dialogo Disposizione form per visualizzare un menu a comparsa

- ✓ Lo stile 2 (Sizable) è lo stile di default. Questo stile visualizza l'icona del menu di controllo, la barra del titolo, i pulsanti di riduzione a icona e di ingrandimento, nonché il pulsante di chiusura. Gli utenti possono spostare, ridimensionare, ingrandire o ridurre a icona questa form.
- ✓ Lo stile 3 (Fixed Dialog) visualizza l'icona del menu di controllo, la barra del titolo e il pulsante di chiusura. Gli utenti possono spostare questa form, ma non possono ridimensionarla, ridurla a icona o ingrandirla.
- ✓ Lo stile 4 (Fixed ToolWindow) visualizza la barra del titolo e il pulsante di chiusura. Gli utenti possono spostare questa form, ma non possono ridimensionarla, ridurla a icona o ingrandirla.
- ✓ Lo stile 5 (Sizable ToolWindow) visualizza la barra del titolo e il pulsante di chiusura. Gli utenti possono spostare e ridimensionare questa form.

Per modificare i bordi attorno alla form mediante la finestra Proprietà, utilizzate la procedura di seguito riportata.

1. **Fate clic su un punto qualsiasi della form dove non sia presente un oggetto.**
2. **Aprite la finestra Proprietà (premete F4).**

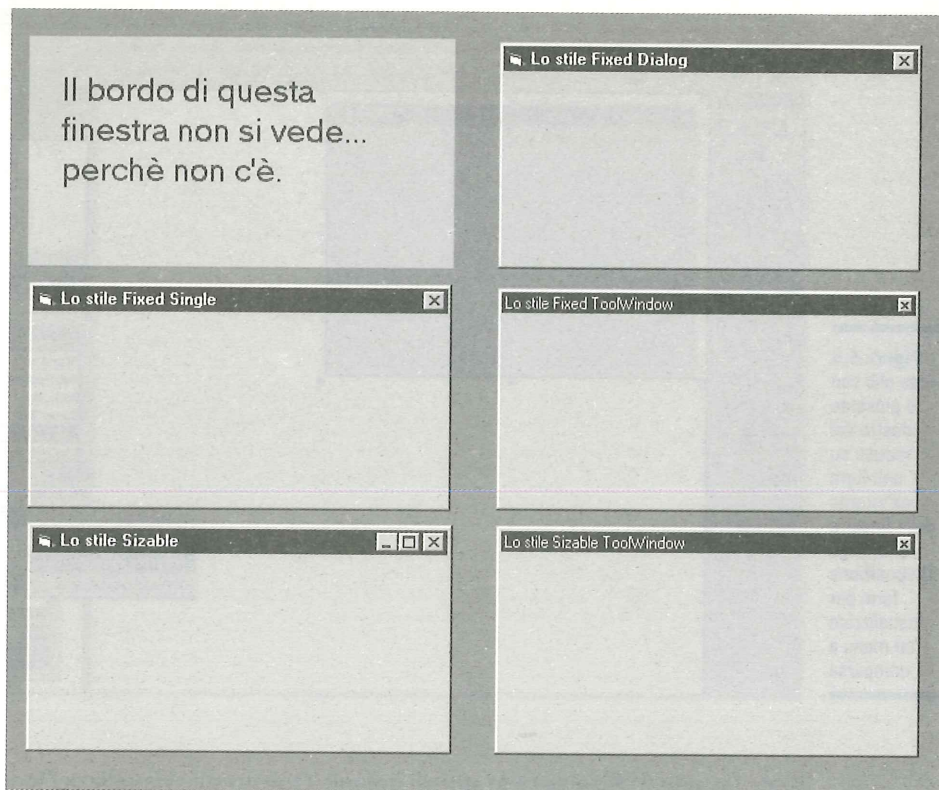



Figura 6.6 I sei diversi stili del bordo per la form

3. **Fate clic sulla proprietà BorderStyle.**
4. **Selezionate una delle seguenti opzioni.**
   - 0 (None)
   - 1 (Fixed Single)
   - 2 (Sizable)
   - 3 (Fixed Dialog)
   - 4 (Fixed ToolWindow)
   - 5 (Sizable ToolWindow)

Ridurre a icona e ingrandire le form

Le form possono occupare parte dello schermo o lo schermo intero. Qualsiasi form che occupi l'intero schermo viene considerata ingrandita. D'altro canto, le

form possono essere rimpicciolite e visualizzate sullo schermo come icone. Una form rimpicciolita viene ridotta a icona. Qualsiasi form che occupi solo una parte dello schermo viene considerata normale. Anche chi pensa che i programmatori inventino troppe definizioni viene considerato normale. Nella Figura 6.7 vengono illustrate le tre dimensioni possibili delle form.

Se alla proprietà BorderStyle attribuite lo stile 0 (None), 3 (Fixed Dialog), 4 (Fixed ToolWindow) o 5 (Sizable ToolWindow) non potete ridurre a icona o ingrandire una form.

Per visualizzare una form come normale, ridotta a icona o ingrandita mentre il programma è in esecuzione, utilizzate la procedura di seguito riportata.

1. **Fate clic su un punto qualsiasi della form (ma non sugli oggetti che essa contiene).**
2. **Premete F4 per aprire la finestra Proprietà.**
3. **Fate clic sulla proprietà WindowState nella finestra Proprietà.**

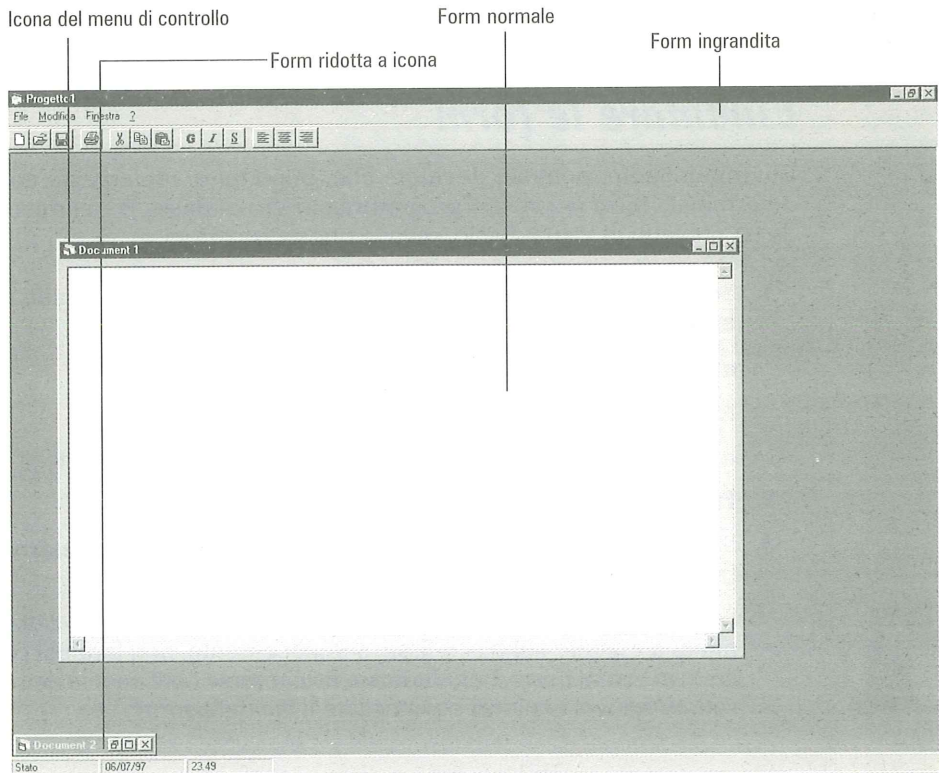



Figura 6.7 Una form ridotta a icona, ingrandita o normale

4. **Selezionate una delle seguenti opzioni.**
   - 0 (Normal)
   - 1 (Minimized)
   - 2 (Maximized)

Inoltre, è possibile offrire all'utente la possibilità di ridurre a icona o ingrandire una form. A questo scopo, sulla vostra form dovranno essere inseriti i pulsanti Minimize (Riduzione a icona) e Maximize (Ingrandimento).

Per visualizzare i pulsanti Minimize e Maximize su una form utilizzate la procedura di seguito riportata.

1. **Fate clic su un punto qualsiasi della form dove non sia presente un oggetto.**
2. **Premete F4 per aprire la finestra Proprietà.**
3. **Fate clic sulla proprietà MinButton (o MaxButton) nella finestra Proprietà e selezionate True o False.**

A seconda dello stile del bordo selezionato per la form, il valore di default per le proprietà MinButton o MaxButton potrebbe essere True o False.

Eliminare le form

Di tanto in tanto, potreste decidere che, dopo tutto, preferireste non avere una determinata form nel vostro programma in Visual Basic. Per eliminare una form da un progetto in Visual Basic, seguite queste istruzioni.

1. **Selezionate una delle possibilità seguenti per passare alla finestra Gestione progetti.**
   - Selezionate Visualizza - Gestione progetti
   - premete Ctrl+R
   - fate clic sull'icona Gestione progetti inclusa sulla barra degli strumenti.
2. **Fate clic sulla form che intendete eliminare.**
3. **Selezionate Project-Remove oppure, con il pulsante destro del mouse, fate clic sul nome della form e selezionate Remove.**

Se eliminate una form che in precedenza avevate salvato, detta form continuerà a esistere sul disco fisso; semplicemente, non farà più parte del progetto in Visual Basic. Per eliminare fisicamente ogni traccia della form, utilizzate Windows Explorer e cancellate il file della form.

Icona del menu di Controllo

Gli utenti possono spostare o ridimensionare una form utilizzando il mouse. Per ridurre a icona o ingrandire una form fate semplicemente clic sul pulsante Riduzione a icona o Ingrandimento. Per gli utenti che non riescono a tenere il passo con la moderna tecnologia, o che si rifiutano di utilizzare il mouse, ogni form fornisce anche un'icona del menu di controllo.

Se scegliete per la proprietà BorderStyle i valori 0 (None), 4 (Fixed ToolWindow) o 5 (Sizable ToolWindow), l'icona del menu di controllo non appare sulle form.

Per attivare l'icona del menu di controllo mentre il programma è in esecuzione, fate clic sull'icona del menu di controllo oppure premete ALT+BARRA SPAZIATRICE. Appare un menu che permette all'utente di spostare, ridimensionare, ridurre a icona e ingrandire una form.

Eliminare un'icona del menu di controllo

L'icona del menu di controllo appare su tutte le form. Tuttavia, se ritenete che questa icona sia veramente brutta e volete eliminarla, utilizzate la procedura di seguito riportata.

1. **Fate clic su un punto qualsiasi della form dove non sia presente un oggetto.**
2. **Premete F4 per aprire la finestra Proprietà.**
3. **Fate clic sulla proprietà ControlBox nella finestra Proprietà.**
4. **Selezionate True o False.**

   A seconda dello stile del bordo selezionato per la form, il valore di default potrebbe essere True o False.

Visualizzare icone come le icone del menu di controllo sulle form

Le icone sono speciali simboli grafici con file caratterizzati dall'estensione ICO. Di solito, Visual Basic visualizza l'icona del menu di controllo di una form con un'icona di default che assomiglia a una bandiera agitata dal vento (Figura 6.7). Se però quest'icona non vi piace, potete personalizzarla. Per modificare l'icona con la finestra Proprietà, seguite i passi illustrati di seguito.

1. **Fate clic su un punto qualsiasi della form, ma non su un oggetto in essa contenuto.**
2. **Aprite la finestra Proprietà (premendo F4).**

3. **Fate doppio clic sulla proprietà Icon nella finestra Proprietà.**

   Visual Basic visualizza così la finestra di dialogo Carica icona.

4. **Selezionate l'icona che intendete utilizzare e fate clic su Apri.**

Quale sarà la form visualizzata per prima da Visual Basic?

Quando il programma è in esecuzione, la prima form visualizzata è la prima che avete creato. Per modificare l'ordine di visualizzazione utilizzate la procedura di seguito riportata.

1. **Selezionate Progetto-Proprietà di NomeProgetto (dove NomeProgetto è il nome del vostro progetto Visual Basic).**

   Visual Basic visualizza la finestra di dialogo Proprietà progetto.

2. **Fate clic sulla scheda Generale.**

   Visual Basic visualizza la scheda illustrata nella Figura 6.8.

3. **Fate clic sulla freccia verso il basso della casella di riepilogo Oggetto di avvio per visualizzare un elenco di tutti i file del vostro progetto.**

4. **Selezionate la form che volete eseguire per prima e fate clic su OK.**

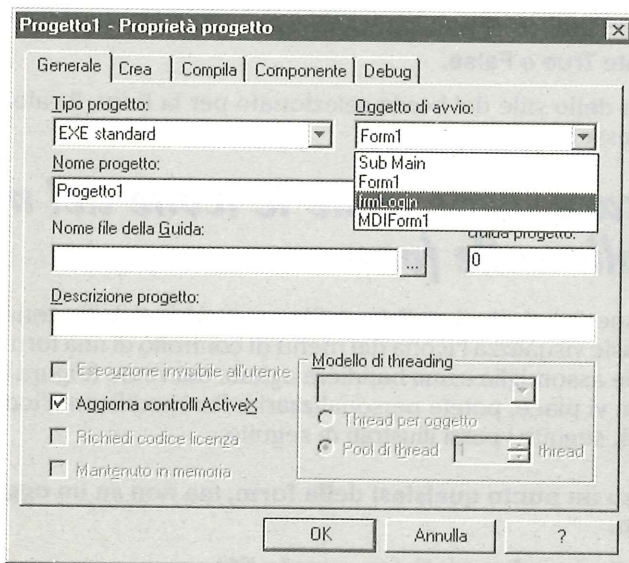



Figura 6.8 La scelta della form da visualizzare per prima nella finestra di dialogo Progetto

Definire le unità di misura su una form

Le form sono utili perché è possibile collocarvi gli oggetti. Per questo motivo, in ogni form è sempre visualizzata una griglia per aiutarvi ad allineare gli oggetti.

Per default, ciascuna form utilizza un'unità di misura definita twip.

Sempre che la cosa possa interessarvi, 1440 twip equivalgono a un pollice. Se non volete utilizzare i twip come unità di misurazione Visual Basic offre sette opzioni.

- ✓ Twip (1440 twip = un pollice)
- ✓ Punti (72 punti = un pollice)
- ✓ Pixel (il numero di pixel necessari per fare un pollice dipende dalla risoluzione del vostro schermo)
- ✓ Caratteri (un carattere è alto 1/6 di pollice e largo 1/12 di pollice)
- ✓ Pollici (1 pollice = un pollice, sorprendente, vero?)
- ✓ Millimetri (25,4 mm = un pollice)
- ✓ Centimetri (2,54 cm = un pollice)

I programmatori intransigenti saranno ben lieti di sapere che Visual Basic permette persino di creare un sistema di coordinate personalizzato. Se la cosa vi interessa, forse fareste meglio a leggere un libro del tipo *Visual Basic per i geni della programmazione* e non questo. Per noi, le sette unità di misura offerte da Visual Basic sono più che sufficienti.

Per modificare la scala della griglia della form utilizzate la procedura di seguito riportata.

1. **Fate clic su un punto qualsiasi della form dove non sia presente un oggetto.**
2. **Aprite la finestra Proprietà (premendo F4).**
3. **Fate clic sulla proprietà ScaleMode.**
4. **Fate clic sulla freccia verso il basso per visualizzare un elenco di tutte le diverse unità di misura che potete utilizzare.**

   Nella Figura 6.9 viene riportato questo elenco.
5. **Selezionate l'unità di misura che volete utilizzare per la form.**

Figura 6.9 La scelta della scala della griglia

Premere i pulsanti

Premere un pulsante è un'azione semplicissima che tutti sono in grado di fare. Persino i bambini sanno premere i pulsanti, il che, per esempio, consente loro di infilare un hot dog in un forno a microonde e gridare di gioia quando esplode sotto i loro occhi. Tutti utilizzano i pulsanti.

La vostra unità disco ha probabilmente un pulsante che premete per espellere un dischetto; il vostro monitor ha un pulsante di accensione e spegnimento. Persino il mouse ha un pulsante (o due o anche tre).

Dato che i pulsanti sono tanto comuni e semplici da utilizzare, i programmi visualizzano spesso pulsanti da premere con il mouse. Invece di costringervi a scorrere numerosi menu per trovare il comando giusto, i pulsanti visualizzano davanti ai vostri occhi tutte le opzioni disponibili. È sufficiente capire quale pulsante premere.

I pulsanti sono una caratteristica di quasi tutti i programmi; ecco perché il resto del capitolo sarà dedicato a come generare, modificare e premere i vostri pulsanti.

Tipi di pulsanti

In linea di principio, un pulsante è solo un'area dello schermo che l'utente può selezionare con un clic del mouse. Quando viene premuto (selezionato con un clic), un pulsante di comando si precipita a eseguire un comando (ecco perché si chiama pulsante di comando). Visual Basic vi permette di creare due tipi di pulsanti: i pulsanti di comando e i pulsanti di immagine.

Un pulsante di comando visualizza un'etichetta oppure un'immagine. Tale etichetta può essere del tutto priva di fantasia come OK, Annulla o Esci, oppure può rappresentare un comando particolare come Cancella file, Pagina successiva o Perdi la prenotazione aerea del Signor Bianchi.

I pulsanti di comando appaiono spesso nelle finestre di dialogo dove il programma visualizza un messaggio tipo: "`Vuoi veramente cancellare la tua dichiarazione dei redditi per evitare conseguenze legali?`". Le opzioni disponibili possono essere Sì e No.

A differenza dei pulsanti di comando che possono visualizzare sia le etichette sia le immagini, un pulsante di immagine può visualizzare soltanto un'immagine. Il vantaggio dei pulsanti di immagine è che possono essere più piccoli dei pulsanti di comando. Lo svantaggio è che se l'utente non sa che cosa rappresenta ciascun pulsante di immagine non saprà neppure utilizzarlo. I pulsanti sono illustrati nella Figura 6.10.

Dato che il significato dei pulsanti di immagine non è immediato come quello dei pulsanti di comando, sarebbe meglio digitare una breve descrizione della funzione del pulsante nella proprietà ToolTipText (consultate il Capitolo 5).

Per esempio, se l'unico sistema per uscire dal vostro programma consiste nel fare clic su un pulsante di immagine che mostra una porta aperta, gli utenti potrebbero non comprendere la relazione e uscire dal programma spegnendo il computer (dopodiché eviteranno accuratamente di utilizzare di nuovo il vostro programma).

Figura 6.10 I pulsanti di comando e i pulsanti di immagine

Creare i pulsanti

Per creare un pulsante di comando utilizzate la procedura di seguito riportata.

1. **Fate clic sull'icona del pulsante di comando nella casella degli strumenti di Visual Basic.**
2. **Posizionate il mouse nel punto della form in cui desiderate disegnare il pulsante di comando.**

3. **Premete e tenete premuto il pulsante sinistro del mouse e trascinate il mouse per creare una casella dedicata al pulsante di comando.**
4. **Rilasciate il pulsante sinistro del mouse per portare a termine l'operazione.**

   Visual Basic visualizza una banalissima etichetta, tipo Command1, sul pulsante di comando.

Quantunque, per gli utenti, i pulsanti di comando siano più semplici da vedere e da utilizzare, potreste voler utilizzare un pulsante di immagine per consentire agli utenti di fare clic su un'immagine per creare programmi multimediali. Per creare un pulsante di immagine utilizzate la procedura di seguito riportata.

1. **Fate clic sull'icona Image nella casella degli strumenti di Visual Basic.**
2. **Trascinate il mouse nel punto della form in cui volete disegnare il pulsante di immagine.**
3. **Premete F4 per aprire la finestra Proprietà.**
4. **Fate doppio clic sulla proprietà Picture.**

   Visual Basic visualizza la finestra di dialogo Carica immagine.
5. **Caricate un file di grafica (bitmap, icona o metafile).**

   Visual Basic utilizzerà questa icona per il pulsante di immagine.

Dopo che avrete creato un pulsante di comando o di immagine, dovrete comunque scrivere il codice BASIC necessario per far sì che il pulsante in questione faccia qualcosa quando l'utente lo seleziona con un clic.

Visualizzare simpatiche icone sui pulsanti di comando

Anche se la maggior parte dei pulsanti di comando si limita a visualizzare un'etichetta, come OK, Annulla, Sì, No o È tutta colpa del passato, potete anche visualizzare le immagini sui pulsanti di comando. Per esempio, invece di avere un pulsante di comando che visualizza semplicemente una dicitura come "OK", potete anche visualizzare l'immagine di un semaforo con la luce verde sul pulsante di comando.

Per fare in modo che un pulsante di comando visualizzi un'icona, è possibile definire le seguenti proprietà.

- ✓ Style (selezionare 1 - grafico)
- ✓ Picture (definisce l'icona che apparirà sul pulsante di comando)

- ✓ DownPicture (definisce l'icona che apparirà quando l'utente farà clic sul pulsante di comando)
- ✓ DisabledPicture (definisce l'icona che apparirà quando il pulsante di comando sarà ombreggiato).

Le proprietà DownPicture e DisabledPicture sono facoltative quando si desidera visualizzare un'icona su un pulsante di comando.

Per visualizzare un'immagine su un pulsante di comando utilizzate la procedura di seguito riportata.

1. **Fate clic sul pulsante di comando sul quale volete visualizzare un'immagine.**
2. **Aprite la finestra Proprietà (premendo F4).**
3. **Fate clic sulla proprietà Style e poi fate clic sulla freccia verso il basso e selezionate 1 - Graphical.**
4. **Fate doppio clic sulla proprietà Picture.**

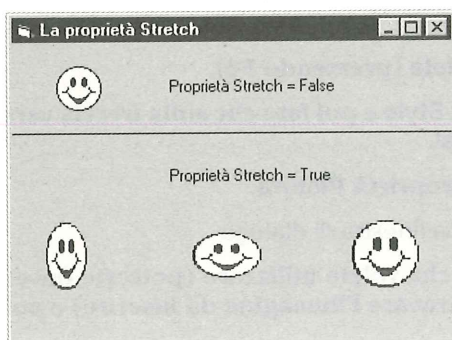
   Visual Basic visualizza una finestra di dialogo.
5. **Fate clic sull'immagine che volete utilizzare (potreste dover esaminare la cartella Graphic per trovare l'immagine da inserire) e poi fate clic su Apri.**
6. **Fate doppio clic sulla proprietà DownPicture per definire l'immagine che deve apparire quando l'utente fa clic sul pulsante di comando.**
7. **Fate clic sull'immagine che volete utilizzare (potreste dover esaminare la cartella Graphic per trovare l'immagine da inserire) e poi fate clic su Apri.**
8. **Fate doppio clic sulla proprietà DisabledPicture per definire l'immagine che deve apparire quando il pulsante di comando viene visualizzato in grigio.**
9. **Fate clic sull'immagine che volete utilizzare (potreste dover esaminare la cartella Graphic per trovare l'immagine da inserire) e poi fate clic su Apri.**

Modificare le dimensioni di un pulsante di immagine

Visual Basic vi permette di disegnare liberamente un riquadro di immagine utilizzando le dimensioni desiderate; tuttavia, se la vostra immagine è troppo grande per il pulsante, tale immagine apparirà tagliata.

Per far sì che le dimensioni dell'immagine grafica cambino ogni volta che modificate le dimensioni del pulsante di immagine, dovete modificare il valore della sua proprietà Stretch impostandolo a True (il valore di default della proprietà Stretch è False).

Per esempio, nella Figura 6.11, il riquadro di immagine nell'angolo in alto a sinistra ha la proprietà Stretch impostata a False quindi, a prescindere dalla larghezza definita per il riquadro di immagine, l'immagine mantiene le stesse dimensioni. Gli altri tre riquadri di immagine hanno la proprietà Stretch impostata a True, di conseguenza, quando vengono modificate le dimensioni del riquadro di immagine anche l'immagine grafica viene adattata.

Figura 6.11 L'effetto della proprietà Stretch sulle dimensioni delle immagini grafiche nei riquadri di immagine

Creare un pulsante di comando di default

Il pulsante di comando di default è quel pulsante che gli utenti possono selezionare semplicemente premendo INVIO non appena viene visualizzato sullo schermo. Il pulsante di default rappresenta la scelta che con maggiore probabilità effettueranno gli utenti.

Per esempio, se l'utente impartisce un comando per lanciare dei missili nucleari contro un'altra nazione, potreste visualizzare una finestra di dialogo che chieda: "Non preferiresti una bella partita a scacchi?" Se il pulsante di default fosse Sì, l'utente potrebbe premere INVIO e, senza rendersene conto, salvare il mondo dal disastro nucleare.

Soltanto i pulsanti di comando possono essere pulsanti di comando di default. I pulsanti di immagine non potranno mai diventare pulsanti di comando di default, semplicemente perché Visual Basic ha deciso così.

Per creare un pulsante di comando di default potete utilizzare uno dei metodi elencati alla pagina successiva.

- ✓ Impostare a 0 la proprietà TabIndex del pulsante di comando; in questo modo il pulsante di comando specificato viene evidenziato come il pulsante di comando di default.
- ✓ Impostare a True la proprietà Default del pulsante di comando di default; questo metodo funziona soltanto se non c'è nessun altro oggetto con una proprietà TabIndex pari a 0.

Per creare un pulsante di comando di default mediante la proprietà TabIndex utilizzate la procedura di seguito riportata.

1. **Fate clic sul pulsante di comando che volete utilizzare come pulsante di comando di default.**
2. **Aprite la finestra Proprietà (premete F4, selezionate Visualizza-Finestra Proprietà oppure fate clic sull'icona Finestra Proprietà nella barra degli strumenti).**
3. **Fate clic sulla proprietà TabIndex e digitate 0.**

Se nessuno degli altri pulsanti di comando presenti sulla form ha la proprietà TabIndex pari a 0, potete creare un pulsante di default impostando la proprietà Default del pulsante di comando a True.

Per creare un pulsante di comando di default mediante la proprietà Default utilizzate la procedura di seguito riportata.

1. **Fate clic sul pulsante di comando che volete trasformare nel pulsante di comando di default.**
2. **Aprite la finestra Proprietà (premete F4).**
3. **Fate clic sulla proprietà Default e impostate il valore a True.**
4. **Controllate che nessun altro oggetto della form abbia una proprietà TabIndex pari a 0.**

Che cosa succede se un pulsante di comando ha la proprietà TabIndex impostata a 0 e un altro pulsante di comando ha la proprietà Default impostata a True? Succede che il pulsante di comando di default sarà quello con la proprietà TabIndex pari a 0. Questo è quanto.

Definire il pulsante Annulla

Quando gli utenti battono con forza su ESC di solito vogliono annullare l'ultimo comando impartito oppure uscire dal programma. Qualsiasi pulsante che consenta all'utente di compiere queste azioni dovrebbe essere definito come pulsante Annulla. Solo un pulsante di comando può essere definito come pulsante Annulla. Un altro bel nome per il pulsante Annulla potrebbe essere Panico, ma que-

sta soluzione distruggerebbe l'idea della programmazione quale scienza pura piuttosto che arte incoerente.

Per creare un pulsante di comando Annulla utilizzate la seguente procedura.

1. **Fate clic sul pulsante di comando che volete trasformare nel pulsante Annulla.**
2. **Aprite la finestra Proprietà (premete F4).**
3. **Fate clic sulla proprietà Cancel, impostate il valore True e modificate l'etichetta del pulsante digitando "Annulla" o qualcosa di simile.**

Raggruppare i pulsanti di comando

Talvolta potreste voler raggruppare una serie di pulsanti di comando correlati per creare un'illusione di organizzazione, come illustrato nella Figura 6.12.

Per creare un gruppo di pulsanti di comando utilizzate la procedura di seguito riportata.

1. **Fate clic sull'icona Frame nella casella degli strumenti di Visual Basic.**
2. **Spostate il mouse nel punto in cui volete disegnare la cornice.**
3. **Tenete premuto il pulsante del mouse e spostate il mouse in modo da disegnare la cornice.**
4. **Fate clic sull'icona del pulsante di comando nella casella degli strumenti di Visual Basic.**
5. **Spostate il mouse all'interno della cornice nel punto in cui volete disegnare un pulsante di comando.**
6. **Tenete premuto il pulsante del mouse e spostate il mouse in modo da disegnare il pulsante di comando all'interno della cornice.**
7. **Ripetete i passi da 4 a 6 fino a quando avrete disegnato tutti i pulsanti di comando desiderati oppure fino a quando deciderete che è ora di fare qualcos'altro.**

Dopo che avrete disegnato un pulsante di comando all'interno di una cornice il pulsante resterà intrappolato per sempre in tale cornice. Quando sposterete la cornice si sposteranno anche tutti i pulsanti di comando in essa contenuti.

Non è possibile creare un pulsante di comando all'esterno di una cornice e poi cercare di spostarlo all'interno di una cornice, quindi non provateci nemmeno.

Se un pulsante di comando risulta essere fuori dalla cornice, selezionate il comando Copy o Cut e fate clic all'interno della cornice in modo da visualizzare le

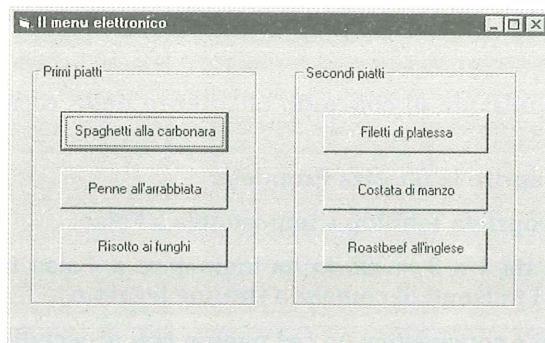



Figura 6.12 Pulsanti di comando raggruppati

maniglie di quest'ultima, quindi selezionate il comando Paste. In questo modo potrete copiare o tagliare un pulsante di comando trasferendolo dall'esterno all'interno della cornice.

Quale ultima modifica per i pulsanti di comando raggruppati impostate a False la proprietà TabStop di tutti i pulsanti raggruppati e poi impostate a True la proprietà TabStop del primo pulsante di comando. In questo modo, se il povero utente dovesse premere TAB verrà evidenziato soltanto il primo pulsante di comando del gruppo.

Mettete alla prova le vostre nuove conoscenze

1. Perché i pulsanti di comando hanno nomi ed etichette?

a. Per offrirvi la doppia possibilità di battezzare un comando con una parolaccia.

b. Il nome è una parolaccia per denominare un pulsante di comando, mentre con l'etichetta potete inserire una battuta divertente.

c. Il nome identifica il pulsante di comando e l'etichetta è ciò che effettivamente appare sullo schermo.

d. Perché lo dice Visual Basic e qualsiasi prodotto venduto da Bill Gates non può sbagliare: Bill Gates è un miliardario.

2. Perché potreste voler raggruppare i pulsanti in una cornice?

a. Per impedire loro di tagliare la corda.

b. Per fare in modo che i comandi correlati siano facilmente individuabili sullo schermo.

c. Per rendere più difficile capire che cosa il vostro programma dovrebbe fare.

d. Per nessuna ragione, se non per creare maggior confusione in coloro che stanno tentando di imparare a programmare un computer per la prima volta.

Per disattivare la proprietà TabStop di un gruppo di pulsanti utilizzate la procedura di seguito riportata.

1. **Fate clic su un pulsante di comando all'interno della cornice, tranne che sul primo.**
2. **Premete F4 per aprire la finestra Proprietà.**
3. **Fate clic sulla proprietà TabStop e impostatela a False.**
4. **Ripetete i passi da 1 a 3 in modo da impostare a False la proprietà TabStop di tutti i pulsanti di comando (tranne il primo).**
5. **Andate in cucina e concedetevi un bel panino con prosciutto e maionese come ricompensa dell'ottimo lavoro svolto.**

Capitolo 7

Compiere delle scelte con caselle e pulsanti

In questo capitolo

- Creare caselle di controllo e pulsanti di opzione
- Creare caselle di riepilogo e caselle combinate
- Ordinare le voci in una casella di riepilogo o in una casella combinata
- Dare un aspetto gradevole alle voci di un elenco

I test a scelta multipla a scuola erano sempre più semplici dei temi, dato che si poteva sempre tirare a indovinare invece di pensare, ottenendo comunque un voto decente. Gli studenti non sono gli unici che preferiscono non pensare se appena possono evitare di farlo; anche la maggior parte degli utenti è della stessa opinione: vuole scelte ordinate davanti ai propri occhi, in modo da poter tentare di indovinare e magari anche riuscirci.

Visual Basic offre diversi metodi per proporre scelte all'utente: caselle di controllo, pulsanti di opzione, caselle di riepilogo e caselle combinate. Le caselle di controllo permettono agli utenti di scegliere una o più opzioni, i pulsanti di opzione permettono di scegliere un'unica opzione, mentre le caselle di riepilogo e le caselle combinate offrono numerose scelte.

Il CD contiene un programma in Visual Basic che illustra il funzionamento di caselle di controllo, pulsante di opzione, caselle di riepilogo e caselle combinate. Sentitevi pienamente liberi di modificare il codice sorgente e vedere cosa accade.

Creare le caselle di controllo e i pulsanti di opzione

Le caselle di controllo (check box) ricordano quegli stupidi questionari che chiedono: "Barrate tutte le caselle che rispondono al vostro caso", come per esempio:

"Perché volete lavorare qui? (Barrate tutte le caselle che rispondono al vostro caso.)"

- ❑ Ho bisogno di soldi
- ❑ Intendo partecipare a un furto organizzato dai dipendenti
- ❑ Desidero un posto dove possa rubare più materiale da ufficio
- ❑ Ho bisogno di un posto sicuro per nascondermi dalla polizia

I pulsanti di opzione (radio button) traggono il loro nome da quelle vecchie autoradio AM che permettevano di cambiare rapidamente stazione premendo un pulsante. Dato che è possibile ascoltare un'unica stazione radio per volta, i pulsanti di opzione vi permettono di selezionare una sola opzione per volta. Ecco un esempio per quanto riguarda i pulsanti di opzione.

Qual è il vostro sesso? (Scegliete una sola voce)

- ❑ Maschile
- ❑ Femminile
- ❑ Ex maschile (femminile grazie alla chirurgia)
- ❑ Ex femminile (maschile grazie alla chirurgia)

Allineare il testo delle caselle e dei pulsanti

Il testo delle caselle di controllo e dei pulsanti di opzione è di solito allineato a sinistra, il che significa che ha questo aspetto:

❑ Questo è allineato a sinistra

Per qualche strana ragione nota soltanto ai pochi programmatori che utilizzano effettivamente le opzioni di allineamento, potete anche allineare a destra il testo delle caselle di controllo e dei pulsanti di opzione:

Questo è allineato a destra ❑

Per allineare a sinistra o a destra il testo di una casella di controllo o di un pulsante di opzione utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di controllo o sul pulsante di opzione che volete allineare.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**

3. **Fate clic sulla proprietà Alignment e impostatela a 0 (Left Justify) oppure a 1 (Right Justify).**

Raggruppare le caselle di controllo

Raramente le caselle di controllo sono isolate. Di solito sono in gruppi di due o più, strette insieme come animali da fattoria spaventati. Il sistema migliore per isolare gruppi di caselle di controllo è utilizzare una cornice. Le cornici separano a livello visivo gruppi diversi di caselle di controllo. Per creare un gruppo di caselle di controllo utilizzate la procedura di seguito riportata.

1. **Fate clic sull'icona Frame nella casella degli strumenti di Visual Basic, come illustrato nella Figura 7.1.**
2. **Spostate il mouse nel punto in cui volete disegnare la cornice.**
3. **Tenete premuto il pulsante sinistro del mouse e muovete il mouse per disegnare una cornice.**
4. **Fate clic sull'icona Check Box nella casella degli strumenti di Visual Basic.**

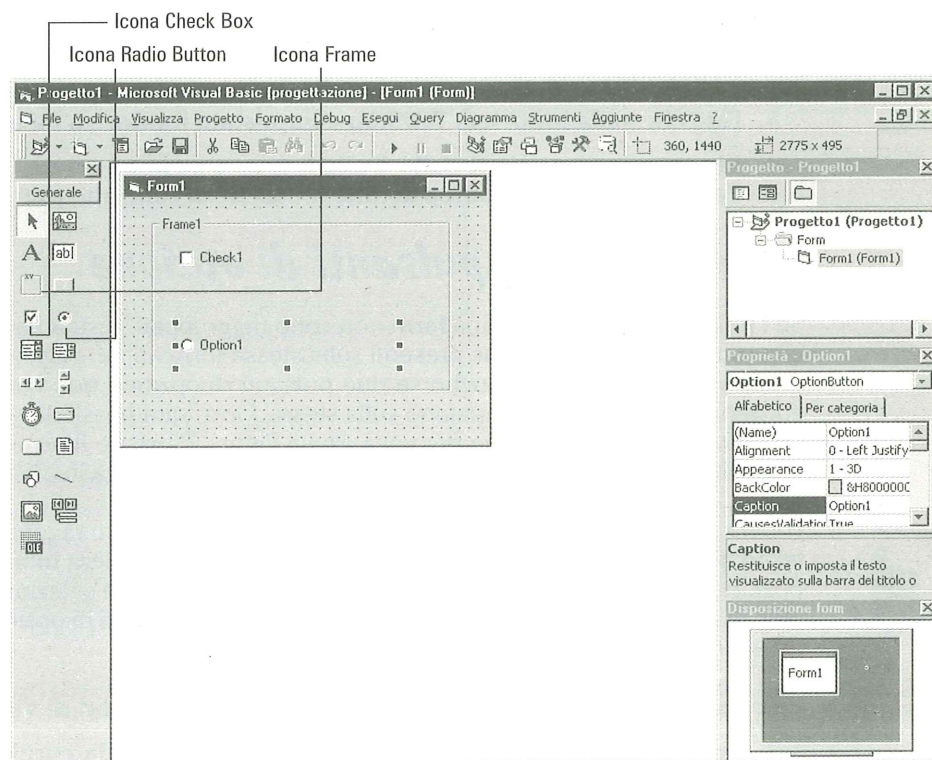



Figura 7.1 Le icone Frame, Check Box e Radio Button della casella degli strumenti di Visual Basic

5. **All'interno della cornice, spostate il mouse nel punto in cui volete disegnare una casella di controllo.**
6. **Tenete premuto il pulsante sinistro del mouse e muovete il mouse per disegnare la casella di controllo.**
7. **Ripetete le operazioni dalla 4 alla 6 fino a quando avrete disegnato tutte le caselle di controllo che volete inserire all'interno della cornice oppure fino a quando deciderete che è ora di concedervi una pausa.**

Quale modifica finale a un gruppo di caselle di controllo, impostate a False le proprietà TabStop di tutte le caselle di controllo raggruppate. Poi impostate la proprietà TabStop della prima casella di controllo a True. In questo modo, quando qualcuno premerà TAB, verrà evidenziata soltanto la prima casella di controllo della cornice.

Per disattivare la proprietà TabStop di un gruppo di caselle di controllo utilizzate la procedura di seguito riportata.

1. **Fate clic su una casella di controllo o su un pulsante di opzione presente all'interno della cornice, tranne che sulla prima.**
2. **Aprite la finestra Proprietà (premete F4, selezionate Visualizza-Finestra Proprietà oppure fate clic sull'icona Finestra Proprietà nella barra degli strumenti).**
3. **Fate clic sulla proprietà TabStop e impostate il valore a False.**
4. **Ripetete le operazioni dalla 1 alla 3 fino a quando avrete impostato a False la proprietà TabStop di tutte le caselle di controllo tranne la prima.**

Raggruppare i pulsanti di opzione

Se i pulsanti di opzione su una form non sono raggruppati, Visual Basic presume che tutti i pulsanti di opzione presenti sulla stessa form appartengano allo stesso gruppo. Di conseguenza, anche se due pulsanti di opzione non hanno nulla in comune tra di loro ma compaiono sulla stessa form, sarà possibile scegliere soltanto un pulsante di opzione per volta. Nella Figura 7.2 viene illustrato come sia possibile selezionare soltanto un pulsante di opzione per volta in ciascun gruppo.

Per visualizzare due o più gruppi di pulsanti di opzione dovete raggrupparli all'interno di una cornice. In caso contrario, Visual Basic raccoglierà tutti i pulsanti di opzione in un unico gruppo, il che significa che sarà possibile selezionare un solo pulsante di opzione per volta. Per creare un gruppo di pulsanti di opzione all'interno di una cornice utilizzate la procedura di seguito riportata.

1. **Fate clic sull'icona Frame nella casella degli strumenti di Visual Basic.**
2. **Spostate il mouse nel punto in cui volete disegnare la cornice.**

Figura 7.2 Il modo in cui i pulsanti di opzione si influenzano a vicenda su una form

3. **Tenete premuto il pulsante sinistro del mouse e muovete il mouse per disegnare una cornice.**
4. **Fate clic sull'icona Radio button nella casella degli strumenti di Visual Basic.**
5. **All'interno della cornice, spostate il mouse nel punto in cui volete disegnare un pulsante di opzione.**
6. **Tenete premuto il pulsante sinistro del mouse e muovete il mouse per disegnare il pulsante di opzione.**
7. **Ripetete le operazioni dalla 4 alla 6 fino a quando avrete disegnato tutti i pulsanti di opzione che volete inserire all'interno della cornice oppure fino a quando non avrete un improrogabile necesità di andare al bagno.**

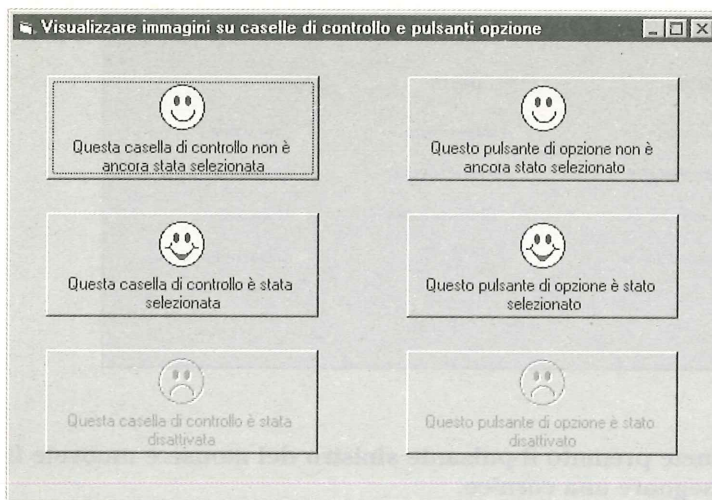
Dopo che avrete disegnato un pulsante di opzione all'interno di una cornice esso rimarrà intrappolato nella cornice per sempre; quando sposterete la cornice si sposteranno anche tutti i pulsanti di opzione.

Aggiungere icone alle caselle di controllo e ai pulsanti di opzione

Solitamente, le caselle di controllo e i pulsanti di opzione visualizzano soltanto un'etichetta, ma se volete essere creativi potete anche visualizzare delle icone sulle caselle di controllo e i pulsanti di opzione.

Per visualizzare sotto forma di icona una casella di controllo o un pulsante di opzione, come mostrato nella Figura 7.3, occorre modificare le seguenti proprietà:

- ✓ **Style:** selezionare 1 - Graphical

Figura 7.3 La visualizzazione delle immagini su una casella di controllo e su un pulsante di opzione

- **Picture:** definisce l'icona che dovrà apparire sulla casella di controllo o sul pulsante di opzione
- **DownPicture:** definisce l'icona che dovrà apparire quando l'utente farà clic sulla casella di controllo o sul pulsante di opzione
- **DisabledPicture:** definisce l'icona che dovrà apparire quando la casella di controllo o il pulsante di opzione sono ombreggiati.

Per qualche strana ragione, Visual Basic fa apparire le caselle di controllo e i pulsanti di opzione con le immagini come se fossero pulsanti di comando, quindi prendete in considerazione questo strano comportamento oppure gli utenti potrebbero scambiare una casella di controllo o un pulsante di opzione per un pulsante di comando.

Per visualizzare un'immagine su una casella di controllo o un pulsante di opzione utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di controllo o sul pulsante di opzione su cui volete visualizzare un'immagine.**
2. **Aprite la finestra Proprietà (premete F4, selezionate Visualizza-Finestra Proprietà oppure fate clic sull'icona Finestra Proprietà nella barra degli strumenti).**
3. **Fate clic sulla proprietà Style e poi fate clic sulla freccia verso il basso e selezionate 1 (Graphical).**
4. **Fate doppio clic sulla proprietà Picture.**

   Visual Basic visualizza una finestra di dialogo.

5. **Fate clic sull'immagine che volete utilizzare (quando la casella di controllo o il pulsante di opzione non è selezionato, potreste dover esaminare la cartella Graphics per trovare l'immagine da inserire) e poi fate clic su Apri.**

6. **Fate doppio clic sulla proprietà DownPicture per definire l'immagine che deve apparire quando la casella di controllo o il pulsante di opzione sono selezionati.**

7. **Fate clic sull'immagine che volete utilizzare (potreste dover esaminare la cartella Graphics per trovare l'immagine da inserire) e poi fate clic su Apri.**

8. **Fate doppio clic sulla proprietà DisabledPicture per definire l'immagine che deve apparire quando la casella di controllo o il pulsante di opzione sono disattivati.**

9. **Fate clic sull'immagine che volete utilizzare (potreste dover esaminare la cartella Graphics per trovare l'immagine da inserire) e poi fate clic su Apri.**

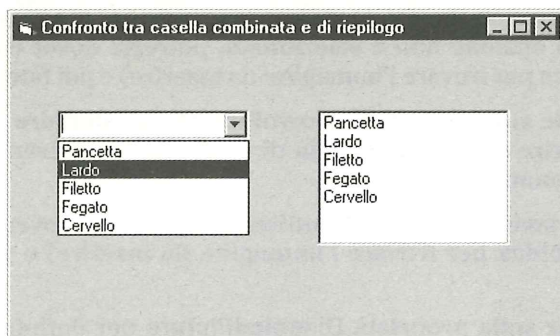
Altre possibilità di scelta con le caselle di riepilogo e le caselle combinate

Quando le scelte sono limitate, le caselle di controllo e i pulsanti di opzione funzionano benissimo. Tuttavia, se avete una decina di opzioni o anche di più, bombardare l'utente con una schermata piena zeppa di caselle di controllo e di pulsanti di opzione può non essere la scelta migliore. Per offrire all'utente molte scelte in modo chiaro e utile, Visual Basic fornisce due alternative alle caselle di controllo e ai pulsanti di opzione: le caselle di riepilogo (list box) e le caselle combinate (combo box). Le caselle di riepilogo visualizzano lunghi elenchi di opzioni tra le quali l'utente può scegliere. Se gli utenti cercano qualcosa che non compare nell'elenco, peggio per loro, non potranno farlo.

Anche le caselle combinate visualizzano lunghi elenchi di opzioni tra le quali l'utente può scegliere; la differenza è che le caselle combinate permettono all'utente di digitare una scelta che non compare nell'elenco. Nella Figura 7.4 viene illustrato un esempio di casella di riepilogo e di casella combinata; notate che quest'ultima visualizza delle voci soltanto se fate clic sulla freccia verso il basso, mentre nella casella di riepilogo le voci sono sempre visualizzate.

Creare caselle di riepilogo e caselle combinate

Le caselle di riepilogo sono simili ai menu dei fast-food, in cui è possibile scegliere solo ciò che compare sul menu perché le persone che lavorano lì non sono in

Figura 7.4 Un confronto tra una casella di riepilogo (sulla destra) e una casella combinata

grado di gestire le richieste particolari. Le caselle combinate sono invece paragonabili ai ristoranti eleganti, dove si può scegliere dal menu oppure dire: "So benissimo che questo è un ristorante vegetariano ma vorrei che il cuoco mi preparasse lo stesso una bistecca alla griglia".

Per creare una casella di riepilogo utilizzate la procedura di seguito riportata.

1. **Fate clic sull'icona List box nella casella degli strumenti di Visual Basic.**
2. **Spostate il mouse nel punto in cui volete disegnare la casella di riepilogo.**
3. **Tenete premuto il pulsante sinistro del mouse e disegnate la casella di riepilogo.**

   Visual Basic visualizza una casella di riepilogo con un titolo banalissimo come List 3.
4. **Ripetete le operazioni dalla 1 alla 3 fino a quando avrete disegnato tutte le caselle di riepilogo di cui avete bisogno.**

Per creare una casella combinata utilizzate la procedura di seguito riportata.

1. **Fate clic sull'icona Combo box nella casella degli strumenti di Visual Basic.**
2. **Spostate il mouse nel punto in cui volete disegnare la casella combinata.**
3. **Tenete premuto il pulsante sinistro del mouse e disegnate la casella combinata.**

   Visual Basic visualizza una casella combinata con un titolo banalissimo come Combo1.
4. **Ripetete le operazioni dalla 1 alla 3 fino a quando avrete disegnato tutte le caselle combinate di cui avete bisogno.**

Gli stili della casella combinata

Una casella combinata vi permette di digitare una scelta oppure di selezionarne una dall'elenco visualizzato. Per aumentare la varietà Visual Basic vi offre tre possibili stili di caselle combinate, come illustrato nella Figura 7.5.

- ✓ Casella combinata a discesa (stile 0 - il valore di default).
- ✓ Casella combinata semplice (stile 1).
- ✓ Casella di riepilogo a discesa (stile 2).

La casella combinata a discesa permette agli utenti di digitare una voce. Se gli utenti non sanno cosa digitare possono fare clic sulla freccia verso il basso posta sul lato destro della casella combinata per visualizzare un elenco di scelte possibili. Visual Basic crea sempre questo tipo di casella combinata, a meno che modifichiate la proprietà Style.

La casella combinata semplice è simile alla casella di riepilogo, ma offre agli utenti anche la possibilità di digitare una voce.

Contrariamente agli altri due stili previsti per le caselle combinate, in questo caso dovrete disegnare con ampie dimensioni una casella combinata in stile semplice in modo che l'utente possa leggere le voci memorizzate.

La casella di riepilogo a discesa è in realtà una casella di riepilogo e visualizza sempre una serie di possibilità, ma non vi permette di digitare nulla di vostra iniziativa.

Se non volete permettere all'utente di scrivere nulla all'interno della casella combinata, impostate la proprietà Style della casella su 2 - Dropdown List.

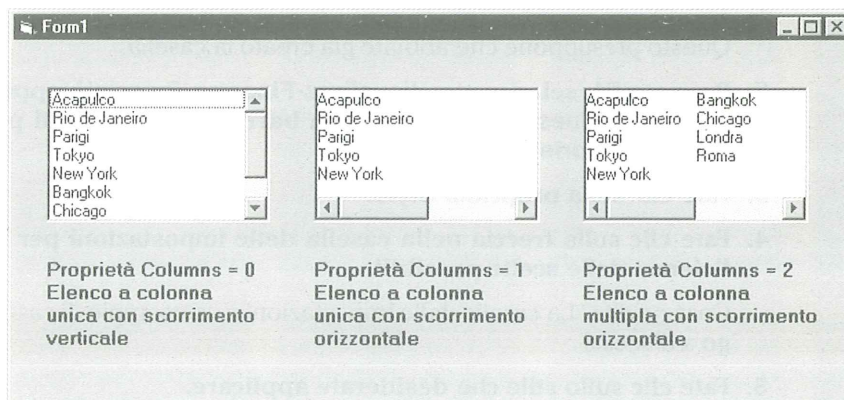



Figura 7.5 I tre stili di una casella combinata

Mettete alla prova le vostre nuove conoscenze

1. **Qual è la differenza principale tra una casella di controllo e un pulsante di opzione?**

   a. È possibile selezionare contemporaneamente una o più caselle di controllo, ma solo un pulsante di opzione.

   b. I pulsanti di opzione servono a sintonizzare la radio sulla stazione preferita, mentre le caselle di controllo servono a riporre gli assegni cancellati.

   c. Chi lo sa è bravo.

   d. Tutto è uguale sotto il sole. Tutto dipende dai punti di vista.

2. **Qual è la differenza principale tra una casella di riepilogo e una casella combinata?**

   a. Una casella combinata offre all'utente la possibilità di digitare una voce o di selezionarne una dall'elenco visualizzato. Una casella di riepilogo lo costringe a scegliere una delle voci dall'elenco visualizzato.

   b. Riepilogo si scrive R-I-E-P-I-L-O-G-O, mentre combinata si scrive C-O-M-B-I-N-A-T-A.

   c. Le caselle combinate sono più belle delle caselle di riepilogo perché tendono a confondere meno l'utente medio.

   d. Non c'è alcuna differenza. Infatti hanno esattamente lo stesso sapore.

A questo punto potreste chiedervi: "Un attimo. Perché dovrei creare una casella combinata per poi trasformarla in una stupida casella di riepilogo?". A differenza delle caselle di riepilogo normali, una casella di riepilogo a discesa non visualizza l'elenco sullo schermo se prima l'utente non fa clic sulla freccia posta sul lato destro della casella. Questo tipo di casella di riepilogo è utile quando volete risparmiare spazio sullo schermo. Per definire lo stile di una casella combinata utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella combinata che volete modificare.**

   Questo presuppone che abbiate già creato la casella.

2. **Premete F4, selezionate Visualizza-Finestra Proprietà oppure fate clic sull'icona Finestra Proprietà nella barra degli strumenti per aprire la finestra Proprietà.**

3. **Fate clic sulla proprietà Style.**

4. **Fate clic sulla freccia nella casella delle impostazioni per visualizzare l'elenco delle scelte possibili.**

   Cosa ne dite? La casella delle impostazioni è un esempio di casella di riepilogo a discesa!

5. **Fate clic sullo stile che desiderate applicare.**

Aggiungere voci alle caselle di riepilogo e alle caselle combinate

Dopo che avete creato la casella di riepilogo o la casella combinata dovrete riempirla di voci (altrimenti che senso avrebbe averla creata?). Visual Basic offre due metodi per aggiungere le voci a queste caselle.

- ✓ Utilizzare la proprietà List nella finestra Proprietà.
- ✓ Utilizzare il codice BASIC.

Per aggiungere le voci a queste caselle mediante la proprietà List utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella alla quale volete aggiungere alcune voci.**
2. **Premete F4, selezionate Visualizza-Finestra Proprietà oppure fate clic sull'icona Finestra Proprietà nella barra degli strumenti per aprire la finestra Proprietà.**
3. **Fate doppio clic sulla proprietà List.**

   Appare una casella di testo a discesa.
4. **Digitate la prima voce che deve apparire nella casella di riepilogo o nella casella combinata e poi premete CTRL+INVIO.**

   Ripetete questa operazione per ogni voce che volete aggiungere.
5. **Premete INVIO.**

Quando digitate voci nella proprietà List, assicuratevi di premere sempre Ctrl+INVIO tra una voce e l'altra. Un INVIO per Visual Basic significa che avete terminato di digitare voci nella proprietà List.

Se volete utilizzare il codice BASIC per aggiungere le voci a una casella di riepilogo o a una casella combinata, il comando segreto da utilizzare è `AddItem`. Quindi, se volete aggiungere la voce "Sceglimi" a una casella di riepilogo denominata cboPrimo ecco il codice BASIC magico che vi permette di farlo:

```
cboPrimo.AddItem "Sceglimi"
```

È possibile aggiungere voci a una casella di riepilogo o a una casella combinata ogni volta che il programma è in esecuzione, sebbene di solito le voci vengano aggiunte quando la form viene caricata per la prima volta.

Per aggiungere le voci a un elenco quando la form viene caricata utilizzate la procedura riportata alla pagina successiva.

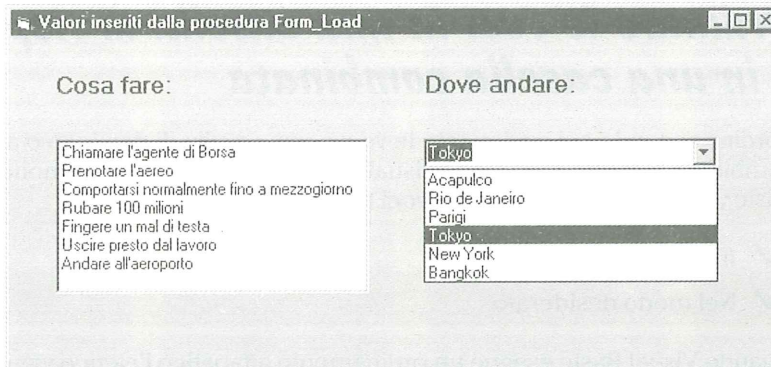
1. **Fate clic sulla form nella finestra Progetto.**
2. **Fate clic sull'icona Visualizza Codice.**
3. **Fate clic sulla casella di riepilogo Oggetto della finestra codice selezionate il nome della form.**

   Visual Basic visualizza la seguente procedura:

```
Private Sub Form_Load()
End Sub
```

4. **Per ciascuna voce che volete visualizzare in una casella di riepilogo o in una casella combinata utilizzate il comando segreto AddItem.**

   Per esempio, se avete una casella di riepilogo denominata 1stAgenda e una casella combinata denominata cboNascondi, la procedura `Sub Form_Load()` potrebbe essere simile alla seguente:

```
Private Sub Form_Load()
  1stAgenda.AddItem "Chiamare l'agente di Borsa"
  1stAgenda.AddItem "Prenotare l'aereo"
  1stAgenda.AddItem "Comportarsi normalmente fino a
  mezzogiorno"
  1stAgenda.AddItem "Rubare 100 milioni"
  1stAgenda.AddItem "Fingere un mal di testa"
  1stAgenda.AddItem "Uscire presto dal lavoro"
  1stAgenda.AddItem "Andare all'aeroporto"
  cboNascondi.AddItem "Acapulco"
  cboNascondi.AddItem "Rio de Janeiro"
  cboNascondi.AddItem "Parigi"
  cboNascondi.AddItem "Tokyo"
  cboNascondi.AddItem "New York"
  cboNascondi.AddItem "Bangkok"
End Sub
```

Questa procedura aggiunge le voci illustrate alla casella di riepilogo 1stAgenda e alla casella combinata cboNascondi ogni volta che viene caricata la prima form del programma; nella Figura 7.6 viene riportato un esempio del risultato finale.

Evidenziare le voci di default

Lo scopo delle caselle di riepilogo e delle caselle combinate è fornire agli utenti delle scelte. Per rendere la selezione delle voci ancora più semplice e quindi più efficace dal punto di vista dell'utente, le caselle combinate possono presentare delle voci di default (nel caso delle caselle di riepilogo la voce di default è la prima). Una *voce di default* è la voce che il computer utilizza quando non riceve istruzioni contrarie; ovviamente, la voce di default deve essere quella più probabile, altrimenti non avrebbe senso.

Figura 7.6 L'aspetto di 1stAgenda e di cboNascondi dopo l'inserimento di alcune voci con la procedura Form_Load

Per creare una voce di default per una casella di riepilogo utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella combinata alla quale volete assegnare una voce di default.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure fate clic sull'icona Finestra Proprietà nella barra degli strumenti.**
3. **Fate clic sulla proprietà Text.**
4. **Digitate la voce che volete far comparire come voce di default su questa casella combinata.**

Se avete impostato la proprietà Style a 2 (DropDown List) potete definire una voce di default utilizzando il codice BASIC per assegnare alla proprietà ListIndex un valore come 0 (per far sì che la prima voce sia quella di default):

```
cboNascondi.ListIndex = 2
```

Quando un utente fa clic su una casella combinata, la voce di default viene evidenziata (Figura 7.6).

Se non definite alcuna voce di default, Visual Basic visualizza il nome generico della casella combinata (Combo1, Combo2 e così via) come voce di default. Dato che questa soluzione è davvero brutta e non si addice a dei professionisti, sarebbe meglio definire sempre una voce di default per le caselle combinate.

Ordinare le voci in una casella di riepilogo o in una casella combinata

L'ordine secondo cui aggiungete le voci a una casella di riepilogo o a una casella combinata è anche l'ordine di visualizzazione. Per non essere monotono Visual Basic vi permette di ordinare le voci nei due modi seguenti.

- ✓ In ordine alfabetico.
- ✓ Nel modo desiderato.

Quando Visual Basic esegue un ordinamento alfabetico l'elenco viene ordinato a prescindere dai caratteri maiuscoli o minuscoli. Per esempio, secondo Visual Basic "Tua Mamma" e "TUA MAMMA" sono assolutamente identici.

Per eseguire un ordinamento alfabetico delle voci presenti in una casella di riepilogo o in una casella combinata utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di riepilogo o sulla casella combinata dove volete eseguire l'ordinamento alfabetico.**
2. **Premete F4, selezionate Visualizza-Finestra Proprietà oppure fate clic sull'icona Finestra Proprietà nella barra degli strumenti per aprire la finestra Proprietà.**
3. **Fate clic sulla proprietà Sorted e impostatela a True.**

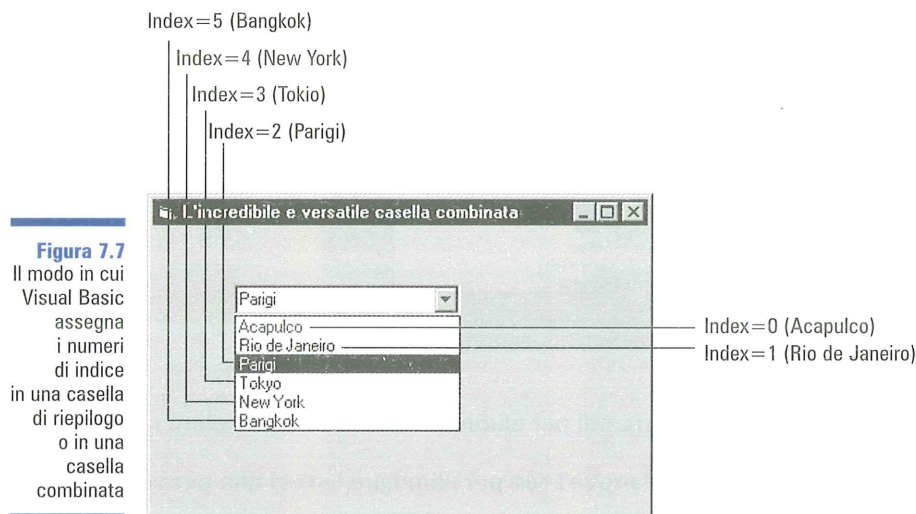
Visual Basic ordina sempre le voci dalla A alla Z; non è possibile eseguire un ordinamento inverso, vale a dire dalla Z alla A (a meno che non capovolgiate il monitor).

Se quello che desiderate non è un ordinamento alfabetico, potete ordinare le voci una per una manualmente. Visual Basic attribuisce un numero di indice (un semplicissimo numero come 1 oppure 3) a ogni voce presente in un elenco.

Alla prima voce dell'elenco viene assegnato il numero 0, alla seconda voce il numero 1, alla terza voce il numero 2 e così via, come illustrato nella Figura 7.7. Se vi è capitato di utilizzare ascensori dove il piano terra viene chiamato 1, il primo piano 2 e il secondo piano 3, capirete quanto il metodo utilizzato da Visual Basic per attribuire i numeri possa essere fonte di confusione.

Per porre un elemento in cima a un elenco è ancora necessario utilizzare il magico comando BASIC `AddItem` in questo modo:

```
cboNascondi.AddItem "Londra", 0
```

Figura 7.7 Il modo in cui Visual Basic assegna i numeri di indice in una casella di riepilogo o in una casella combinata

Se omettete il numero di indice, Visual Basic aggiunge le voci all'elenco in uno dei due modi seguenti.

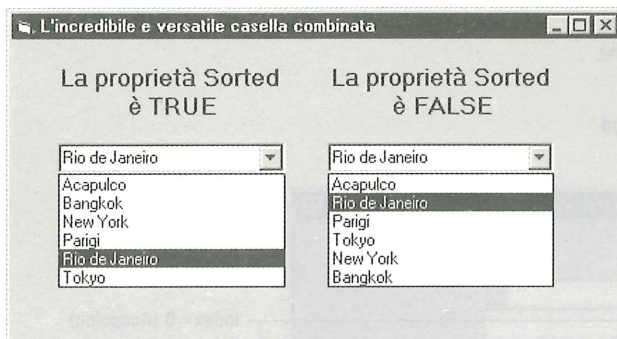
- ✓ Se la proprietà Sorted della casella di riepilogo o della casella combinata è impostata a False, la voce viene inserita in fondo all'elenco.
- ✓ Se la proprietà Sorted della casella di riepilogo o della casella combinata è impostata a True, la voce viene inserita secondo l'ordinamento alfabetico corretto.

Nella Figura 7.8 viene mostrato che, se omettete il numero di indice, la voce "Rio de Janeiro" viene inserita in ordine alfabetico nell'elenco ordinato e al secondo posto nell'elenco non ordinato.

Se impostate a True la proprietà Sorted di una casella di riepilogo o di una casella combinata e aggiungete le voci utilizzando i numeri di indice, Visual Basic aggiunge le voci in base ai numeri di indice e non esegue l'ordinamento alfabetico.

Cancellare delle voci da una casella di riepilogo o da una casella combinata

Aggiungere voci e ordinarle può abbellire i vostri elenchi ma, talvolta, è più divertente cancellare una voce per soddisfare l'impulso distruttivo che tutti noi proviamo di tanto in tanto.

Figura 7.8 L'aggiunta di una voce in un elenco ordinato alfabeticamente e in uno non ordinato alfabeticamente

Visual Basic offre due comandi per eliminare una voce da un elenco.

- ✓ Il comando BASIC `RemoveItem` per eliminare le voci una per volta.
- ✓ Il comando BASIC `Clear` per eliminare un intero elenco in un colpo solo.

Per utilizzare il comando BASIC `RemoveItem` dovete conoscere il numero di indice dell'elemento che volete eliminare. Per esempio, per eliminare la voce con il numero di indice 5 presente in una casella di riepilogo denominata 1stAgenda, utilizzate il seguente comando BASIC:

```
1stAgenda.RemoveItem 5
```

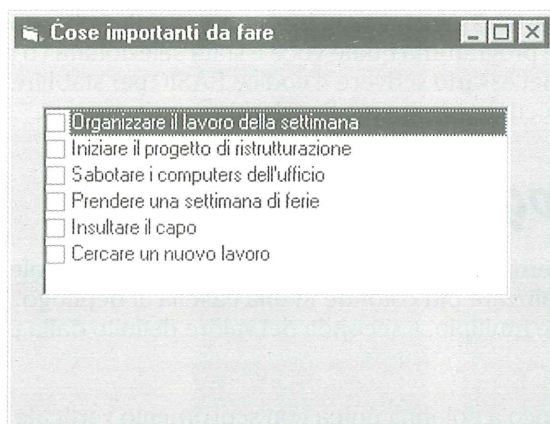
Se volete utilizzare il comando `Clear` per eliminare un intero elenco in un sol colpo, dovete utilizzare il nome della casella di riepilogo o della casella combinata che contiene l'elenco che volete eliminare. Per esempio, per eliminare l'intero contenuto di una casella combinata denominata cboNascondi utilizzate il seguente comando BASIC:

```
cboNascondi.Clear
```

Prima di utilizzare il comando BASIC `Clear`, accertatevi di voler eliminare davvero l'intero elenco.

Visualizzare le caselle di controllo nelle caselle di riepilogo

Quando visualizzate le voci in una casella di riepilogo potete anche visualizzarle come caselle di controllo (Figura 7.9). Se utilizzate le caselle di controllo potete spuntare le voci in una casella di riepilogo che contiene gli elenchi di cose da fare, gli elenchi del droghiere oppure un elenco di buoni propositi che non avete intenzione di rispettare.

Figura 7.9 La visualizzazione delle caselle di controllo in una casella di riepilogo

Per visualizzare le caselle di controllo in una casella di riepilogo utilizzate la procedura di seguito riportata.

1. **Fate clic nella casella di riepilogo in cui volete far apparire le caselle di controllo.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure fate clic sull'icona Finestra Proprietà nella barra degli strumenti.**
3. **Fate clic sulla proprietà Style e selezionate 1 (Checkbox).**

Per capire quale casella di controllo ha scelto l'utente nella lista di riepilogo, dovrete utilizzare il codice BASIC. Ricordate che a ciascuna voce delle caselle di riepilogo viene attribuito un numero. Alla prima voce dell'elenco viene attribuito il numero 0, alla seconda il numero 1, alla terza il numero 2 e così via. Detto ciò, occorre scrivere una procedura di evento che vi dica quale casella è stata selezionata (o deselezionata). A questo scopo, dovrete servirvi della procedura di evento collegata alla casella di riepilogo, tipo:

```
Private Sub List1_ItemCheck (Item as Integer)
GlobalVariable = Item

' Questo stamperà la casella selezionata (o deselezionata)
Print Item
End Sub
```

Nel momento in cui qualcuno selezionerà (o deselezionerà) una casella di controllo della casella di riepilogo, la procedura di evento ItemCheck verrà eseguita e la variabile Item vi dirà quale voce è stata selezionata (o deselezionata). Quindi, se viene selezionata (o deselezionata) la prima voce dell'elenco, la procedura di evento ItemCheck attribuirà ad Item il numero 0.

A questo punto dovrete attribuire ad Item la vostra variabile in modo da comunicare alle altre parti del programma quale voce è stata selezionata (o deselezionata). Dopodiché, sarà necessario scrivere il codice BASIC per stabilire cosa debba accadere quando qualcuno seleziona o deseleziona una voce.

Caselle di riepilogo a più colonne

Per motivi estetici, o semplicemente perché vi state annoiando e volete pasticciare un po', potete visualizzare più colonne in una casella di riepilogo. Visual Basic offre tre tipi di colonne multiple, a seconda del valore definito dalla proprietà Column.

- ✓ **Valore 0:** un elenco a colonna unica con scorrimento verticale (l'aspetto di default delle caselle di riepilogo).
- ✓ **Valore 1:** un elenco a colonna singola con scorrimento orizzontale (ma senza possibilità di scorrimento verticale).
- ✓ **Qualsiasi valore superiore a 1:** un elenco a colonna multipla con due o più colonne a scorrimento orizzontale (ma senza possibilità di scorrimento verticale).

Questi stili sono illustrati nella Figura 7.10. Per modificare il tipo di visualizzazione delle voci sulle colonne utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di riepilogo che volete modificare.**
2. **Aprite la finestra Proprietà.** A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure fate clic sull'icona Finestra Proprietà nella barra degli strumenti.
3. **Fate clic sulla proprietà Columns e digitate 0, 1 o un qualsiasi numero superiore a 1.**

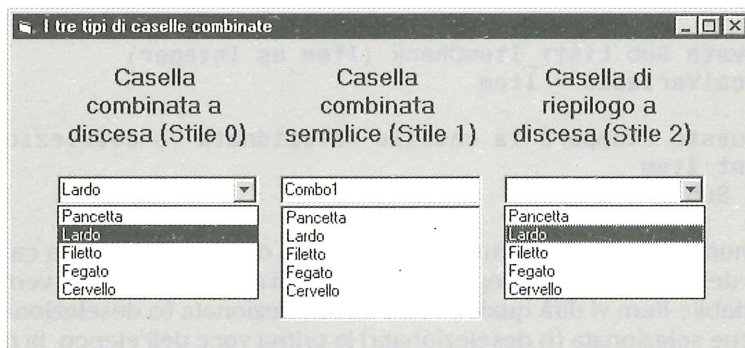



Figura 7.10 Il confronto tra i tre stili di una casella di riepilogo

Dare un aspetto gradevole alle voci di un elenco

Per dare agli elenchi un aspetto più piacevole e renderli meno simili a banalissime liste della spesa Visual Basic vi consente di modificare il tipo di carattere, lo stile del carattere e le dimensioni delle voci dell'elenco.

I tipi di carattere (font) sono modi diversi di visualizzare un testo. Solitamente, Visual Basic utilizza il tipo di carattere MS Sans Serif, ma voi potete utilizzare qualsiasi tipo di carattere installato sul vostro computer (l'MS Sans Serif è simile al tipo di carattere Helvetica, mentre il tipo di carattere MS Sans Serif di Visual Basic è simile al Times Roman).

Per modificare i tipi di carattere delle voci presenti in una casella di riepilogo o in una casella combinata utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di riepilogo o sulla casella combinata di cui volete modificare il tipo di carattere.**
2. **Premete F4, selezionate Visualizza-Finestra Proprietà oppure fate clic sull'icona Finestra Proprietà nella barra degli strumenti per aprire la finestra Proprietà.**
3. **Fate doppio clic sulla proprietà Font.**

   Visual Basic visualizza una finestra di dialogo Carattere.
4. **Fate clic sul tipo di carattere desiderato e poi fate clic su OK.**

   Visual Basic modifica immediatamente il tipo di carattere.

Fate attenzione quando utilizzate i tipi di carattere. I principianti si lasciano spesso prendere dalla foga e utilizzano una quantità eccessiva di tipi di carattere bizzarri. Se non avete una buona ragione per utilizzare tipi di carattere diversi, lasciate che Visual Basic utilizzi il tipo di carattere di default, ovvero MS Sans Serif.

Potete anche modificare le dimensioni delle voci, rendendole più grandi o più piccole. Tuttavia, più le dimensioni dei caratteri sono grandi, più grande dovrà essere la casella di riepilogo o la casella combinata per visualizzare le voci senza troncarle.

Più le dimensioni del tipo di carattere sono grandi, più le voci della casella di riepilogo o della casella combinata risulteranno evidenti. Solitamente, sarebbe meglio utilizzare un'unica dimensioni per tutte le caselle di riepilogo o le caselle combinate, in modo da evitare di confondere l'utente più del necessario.

Per definire le dimensioni del tipo di carattere utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di riepilogo o sulla casella combinata che volete modificare.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure fate clic sull'icona Finestra Proprietà nella barra degli strumenti.**
3. **Fate doppio clic sulla proprietà Carattere.**

   Visual Basic visualizza una finestra di dialogo Font.
4. **Fate clic sulle dimensioni del tipo di carattere che desiderate e poi fate clic su OK.**

   Visual Basic modifica immediatamente le dimensioni del tipo di carattere.

Oltre a modificare il tipo di carattere e le dimensioni potete anche modificare lo stile del tipo di carattere e visualizzare il testo in **grassetto**, *corsivo*, sottolineato o ~~barrato~~. Per impostare uno o più stili utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di riepilogo o sulla casella combinata che volete modificare.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure fate clic sull'icona Finestra Proprietà nella barra degli strumenti.**
3. **Fate doppio clic sulla proprietà Font.**

   Visual Basic visualizza una finestra di dialogo Carattere.
4. **Fate clic sullo stile del tipo di carattere e poi fate clic su OK.**

   Visual Basic modifica immediatamente lo stile del tipo di carattere.

Più le vostre caselle di riepilogo e caselle combinate sono attraenti, maggiori saranno le probabilità che l'utente perlomeno le noti (anche se non le utilizza). Ricordate che il vostro programma deve essere semplice da utilizzare, non un'opera d'arte. Se volete fare i creativi, divertitevi a dipingere con le dita. Se volete creare programmi utili e diventare milionari, rendete i vostri programmi semplici, divertenti e facili da utilizzare.

Capitolo 8

Caselle di testo per digitare e visualizzare parole

In questo capitolo

- Creare le caselle di testo
- Riempire le caselle di testo e nascondere le password
- Utilizzare diversi tipi di carattere, dimensioni, stili e colori

Sebbene le icone e le interfacce grafiche utente siano sempre più amate, non è possibile presentare tutte le scelte utilizzando i pulsanti di comando, i pulsanti di opzione o le caselle combinate. Talvolta, il vostro programma potrebbe dover visualizzare sullo schermo una parola, una frase, un paragrafo o un romanzo. E talvolta anche l'utente potrebbe voler digitare una parola o due.

Qual è la soluzione? Le caselle combinate funzionano bene con le parole o le frasi brevi, ma se il programma deve visualizzare un brano di testo o l'utente deve digitare una quantità notevole di informazioni, una *casella di testo* può rendere molto più semplice questo compito.

Le caselle di testo hanno un duplice scopo:

- ✓ visualizzare il testo sullo schermo;
- ✓ permettere all'utente di digitare un testo.

Le caselle di testo sono uno degli oggetti di programmazione più flessibili poiché possono essere utilizzate per visualizzare istruzioni e l'utente può inserire una risposta a un commento. Se nei vostri programmi utilizzerete una quantità sufficiente di caselle di testo, potreste anche contribuire ad elevare il livello di alfabetizzazione della popolazione generale.

Il programma in Visual Basic incluso sul CD-ROM offre un semplice esempio di come sia possibile utilizzare una casella di testo per accettare una password. Tuffatevi nel codice sorgente ed esercitatevi. Al massimo, potrete rovinare il programma, tuttavia nel frattempo avrete comunque imparato qualcosa.

Creare una casella di testo

Le caselle di testo sono simili a piccoli programmi di elaborazione testi che però possono visualizzare il testo soltanto con un unico tipo di carattere, un'unica dimensione e un unico stile (come il grassetto e il corsivo). Quindi, anche se vi piace utilizzare molti tipi di carattere diversi in una casella di testo, abbandonate immediatamente questa idea perché non potete farlo.

Quando l'utente digita un testo in una casella di testo può utilizzare i seguenti tasti:

- ✓ **CANC:** cancella il carattere a destra del cursore;
- ✓ **BACKSPACE:** cancella il carattere a sinistra del cursore;
- ✓ **MAIUSC+TASTO FRECCIA:** evidenzia un blocco di testo;
- ✓ **CTRL+←:** sposta il cursore di una parola a sinistra;
- ✓ **CTRL+→:** sposta il cursore di una parola a destra;
- ✓ **↖ (o CTRL+HOME):** sposta il cursore all'inizio della riga;
- ✓ **FINE (o CTRL+Fine):** sposta il cursore alla fine della riga;
- ✓ **MAIUSC+qualsiasi tasto di movimento (come Home):** evidenzia il testo;
- ✓ **F11 o F12:** non fanno assolutamente nulla e sono utili quanto i denti del giudizio.

Per creare una casella di testo utilizzate la procedura di seguito riportata.

1. **Fate clic sull'icona Text box (casella di testo) nella casella degli strumenti di Visual Basic.**
2. **Sulla form, spostate il mouse nel punto in cui volete inserire la casella di testo.**
3. **Tenete premuto il pulsante sinistro del mouse e disegnate la casella di testo.**

   Visual Basic visualizza la casella di testo con il testo di default, come Text1.
4. **Ripetete le operazioni dalla 1 alla 3 fino a quando avrete disegnato tutte le caselle di testo desiderate oppure fino a quando troverete qualcosa di più interessante da fare.**

Aggiungere dei graziosi bordi attorno alle caselle di testo

Visual Basic visualizza solitamente attorno alle caselle di testo una riga singola che definisce i confini della casella. Se volete costringere gli utenti a indovinare

dove si trova la casella di testo, potete cancellare questo bordo. Nella Figura 8.1 viene illustrata una casella di testo con il bordo e una senza bordo.

Per modificare i bordi attorno a una casella di testo utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di testo della quale volete modificare il bordo.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate clic sulla proprietà BorderStyle e selezionate una delle seguenti opzioni:**
   - 0 (None);
   - 1 (Fixed Single).

Visualizzare delle parole in una casella di testo

Dopo che avete creato la casella di testo, la fase successiva consiste nell'inserire una parte di testo. Per default, Visual Basic visualizza nelle caselle di testo il nome della casella stessa, come `Text 1`.

Dato che la modifica della proprietà Text di una casella di testo non influisce sulla proprietà Name della casella, se volete che la casella di testo visualizzi qualcosa di più eccitante di Text1 dovete modificare la proprietà Text.

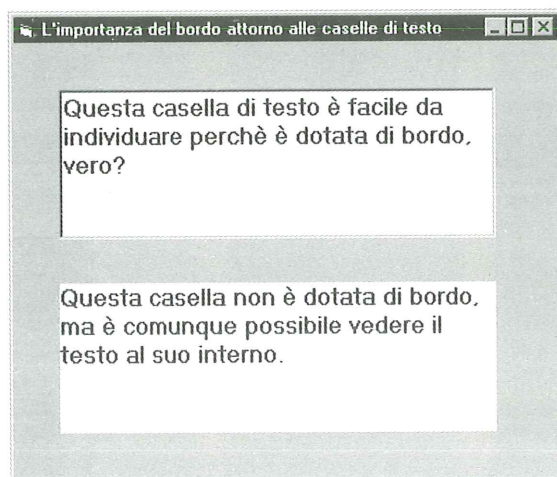



Figura 8.1 Una casella di testo con bordo e una senza bordo

La proprietà Text può contenere di tutto, da una riga vuota (il che significa che la casella di testo sullo schermo sarà vuota) a un normale testo, a un ammasso di scarabocchi incomprensibili che assomigliano a un tipico manuale di informatica. La Figura 8.2 mostra in che modo la proprietà Text determina cosa debba apparire all'interno di una casella di testo. Per modificare la proprietà Text di una casella di testo utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di testo della quale desiderate modificare la proprietà Text.**
2. **Premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti per aprire la finestra Proprietà.**
3. **Fate doppio clic sulla proprietà Text (oppure fate clic sull'etichetta Text) e digitate il testo che volete visualizzare nella casella di testo.**

Se volete modificare il contenuto della casella di testo mentre il programma è in esecuzione dovete utilizzare una linea di codice BASIC analoga alla seguente:

```
txtMessage.Text = "Questo testo verrà visualizzato nella
casella di testo."
```

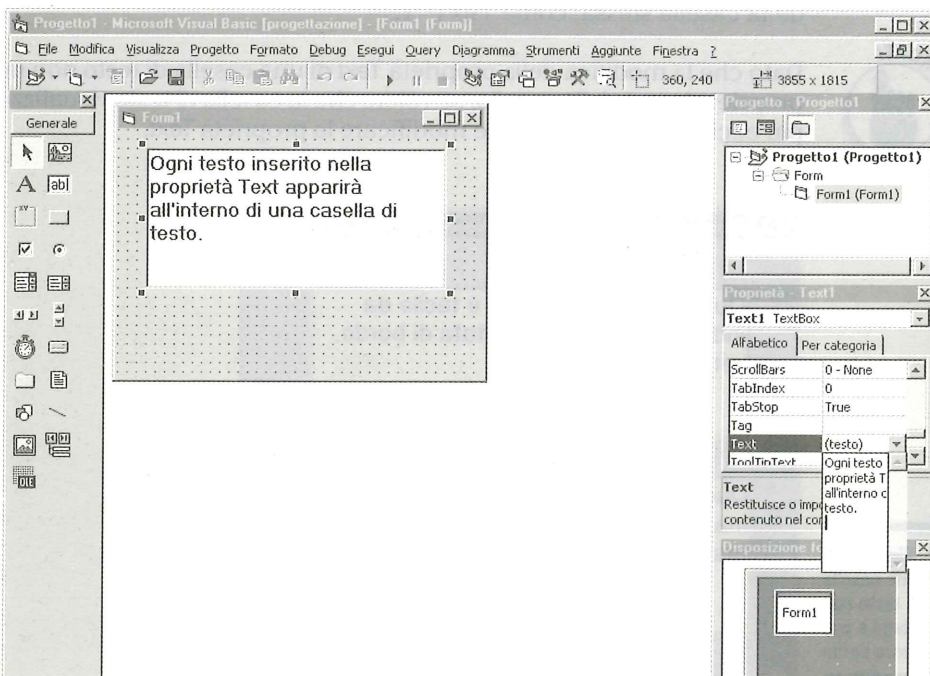



Figura 8.2 Come appare il contenuto della proprietà Text all'interno di una casella di testo

Ogniqualvolta modificate la proprietà Text di una casella di testo il nuovo contenuto cancellerà completamente il precedente contenuto della casella di testo.

Allineare il testo in una casella di testo

Per dare alla vostra casella di testo un aspetto gradevole e organizzato, Visual Basic vi permette di allineare il testo a sinistra, a destra oppure di centrarlo, come illustrato nella Figura 8.3.

Per allineare il testo in una casella di testo utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di testo della quale volete allineare il testo.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate clic sulla proprietà MultiLine e impostate il valore a True. Se la proprietà MultiLine è impostata a False Visual Basic ignora le modifiche apportate alla proprietà Alignment.**
4. **Fate clic sulla proprietà Alignment e selezionate una delle seguenti opzioni:**
   - 0 (Left Justify);
   - 1 (Right Justify);
   - 2 (Center).

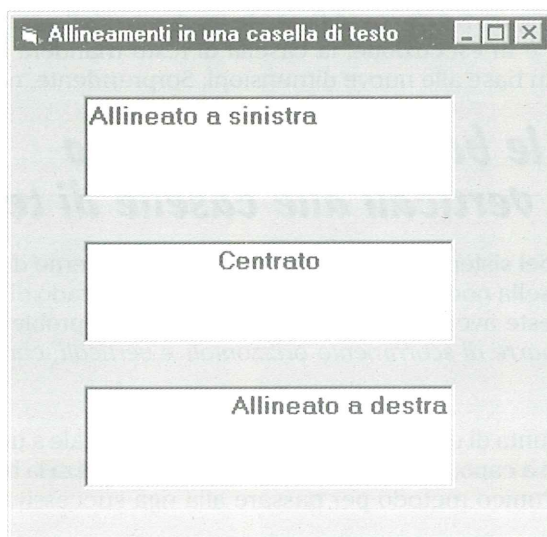



Figura 8.3 Un confronto tra un testo allineato a sinistra, uno allineato a destra e uno centrato

Il fatto di modificare la proprietà Alignment non determina l'allineamento del testo se prima la proprietà MultiLine della casella di testo non viene impostata su True.

Mandare a capo le parole nelle caselle di testo

Le caselle di testo, oltre a visualizzare il testo che il programmatore ha digitato nella proprietà Text, permettono anche agli utenti di digitare un proprio testo. Per default, una casella di testo si comporta in modo abbastanza stupido nella gestione del testo. Se digitate un testo, la casella di testo lo visualizza su un'unica riga che scorre verso sinistra quando si raggiunge il margine destro. Per fare in modo che una casella di testo adotti il ritorno a capo automatico come se fosse un vero programma di elaborazione testi, dovete impostare a True la proprietà MultiLine della casella di testo.

Per impostare a True la proprietà MultiLine utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di testo nella quale desiderate attivare il ritorno a capo.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate clic sulla proprietà MultiLine e impostate il valore a True.**

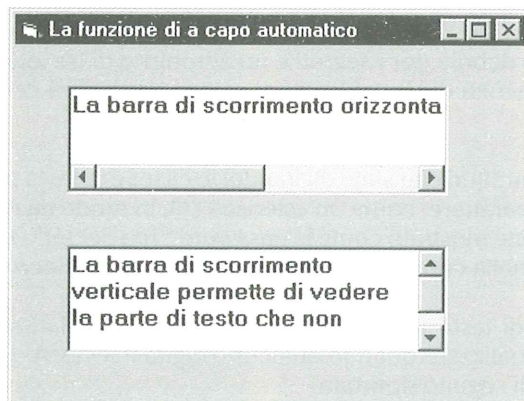
Quando la proprietà MultiLine di una casella di testo è impostata a True, la casella di testo attiva il ritorno a capo automatico rispettando i limiti della casella stessa, come illustrato nella Figura 8.4. Se modificate la larghezza della casella di testo mentre il programma è in esecuzione, la casella di testo manderà automaticamente a capo il testo in base alle nuove dimensioni. Sorprendente, no?

Aggiungere le barre di scorrimento orizzontali e verticali alle caselle di testo

Il ritorno a capo è un bel sistema per visualizzare il testo all'interno di una casella di testo, ma se tale casella non è abbastanza alta non sarà in grado di visualizzare tutto il testo che potreste avere digitato. Per risolvere questo problema potreste dover aggiungere le *barre di scorrimento orizzontali* e *verticali*, come illustrato nella Figura 8.5.

Fate attenzione! L'aggiunta di una barra di scorrimento orizzontale a una casella di testo disattiva il ritorno a capo; quando una casella di testo utilizza la barra di scorrimento orizzontale, l'unico metodo per passare alla riga successiva è premere INVIO.

Figura 8.4 Il ritorno a capo automatico in una casella di testo

Figura 8.5 Le barre di scorrimento orizzontali e verticali nelle caselle di testo

Aggiungendo una barra di scorrimento verticale l'utente potrà digitare e visualizzare un testo più lungo di quello visualizzabile dalla casella. Gli utenti possono inoltre premere ↑ e ↓ per visualizzare il testo in una casella che utilizza la barra di scorrimento verticale.

Per aggiungere le barre di scorrimento a una casella di testo utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di testo nella quale desiderate aggiungere le barre di scorrimento.**
2. **Premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti per aprire la finestra Proprietà.**

3. **Fate clic sulla proprietà ScrollBars e selezionate una delle seguenti opzioni:**
   - 0 (None);
   - 1 (Horizontal);
   - 2 (Vertical);
   - 3 (Both horizontal and vertical).

Le barre di scorrimento orizzontali e verticali funzionano solo se avete impostato a True la proprietà MultiLine della casella di testo (non avrebbe molto senso disporre di barre di scorrimento quando non è possibile visualizzare più di una riga di testo).

Creare una casella di testo per password

Nel caso lavoriate per la CIA, l'FBI, il KGB, la DIA o qualsiasi altra organizzazione con moltissimi soldi, un debole per i segreti e un acronimo di tre lettere, potreste essere interessati alla facoltà di Visual Basic di creare particolari caselle di testo per password.

Invece di visualizzare i caratteri che state digitando, le caselle di testo per password visualizzano un singolo carattere, come un asterisco (*), in modo da mascherare il testo. Nella Figura 8.6 viene mostrato come la password "Top Secret" venga visualizzata solo con asterischi nella casella di testo "CIA Software (Top Secret)".

Per creare una casella di testo per password dovete definire il carattere che la casella di testo deve visualizzare quando qualcuno digita il testo. A questo scopo, utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di testo che volete trasformare in una casella di testo per password.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate clic sulla proprietà MultiLine e impostate il valore a False.**

Impostate sempre la proprietà MultiLine di una casella di testo destinata a una password su False, altrimenti sarà impossibile mascherare i caratteri digitati. Questo è il modo adottato da Visual Basic per dirvi che le password non possono essere così lunghe da richiedere due o più righe.

4. **Fate clic sulla proprietà PasswordChar e digitate il carattere di mascheramento, per esempio un asterisco.**

Il valore inserito per la proprietà PasswordChar deve essere costituito da un singolo carattere.

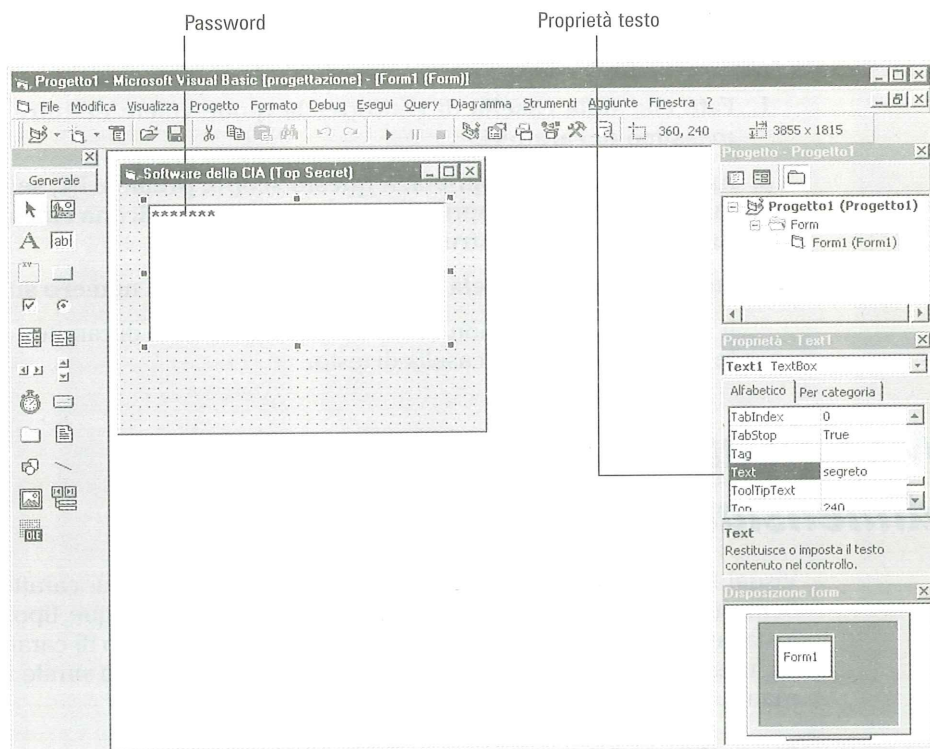



Figura 8.6 Come funziona una casella di testo per password

Limitare la lunghezza del testo

Per evitare che la gente diventi troppo prolissa, potete impostare una lunghezza massima del testo da inserire nella casella di testo. In questo modo potrete impedire agli utenti di digitare racconti incoerenti su quello che hanno fatto l'estate scorsa.

Per definire il numero massimo di caratteri che una casella di testo accetterà da un utente modificate la proprietà MaxLength. Se l'utente tenterà di digitare dei caratteri oltre il limite MaxLength, Visual Basic emetterà un segnale acustico e non ne accetterà più.

Visual Basic non possiede però una proprietà che definisca la lunghezza minima. Per le normali caselle di testo questo non costituisce un problema, ma se state creando una casella di testo per password, sicuramente ci sarà almeno un utente sprovveduto che sceglierà una password composta da un'unica lettera, che quindi potrà essere immediatamente identificata da qualche hacker.

Per definire il numero massimo di caratteri che una casella di testo può accettare utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di testo per la quale volete definire un numero massimo di caratteri.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate clic sulla proprietà MaxLength e digitate un numero superiore a 0.**

   Il valore 0 indica che non esiste un limite al numero di caratteri che un utente può digitare in una casella di testo.

Modificare tipi di carattere, dimensioni e stili

Visual Basic visualizza in genere il testo utilizzando il tipo di carattere MS Sans Serif, ma se volete essere creativi potete scegliere qualunque tipo di carattere installato sul vostro computer (l'MS Sans Serif è simile al tipo di carattere Helvetica, mentre il tipo di carattere MS Sans Serif di Visual Basic è simile al Times Roman).

Per modificare il tipo di carattere in una casella di testo utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di testo della quale desiderate modificare il tipo di carattere.**
2. **Premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti per aprire la finestra Proprietà.**
3. **Fate clic sulla proprietà Font.**

   Visual Basic visualizza una finestra di dialogo Carattere.

4. **Fate clic sul tipo di carattere desiderato e poi fate clic su OK.**

   Visual Basic modifica immediatamente il tipo di carattere nella casella di testo.

Fate attenzione quando utilizzate i tipi di carattere. I principianti spesso si lasciano prendere la mano e scelgono strani tipi di carattere che confondono le idee. Se non avete una ragione davvero valida per modificare il tipo di carattere, continuate a utilizzare il tipo di carattere di default, ossia l'MS Sans Serif, per la maggior parte del vostro lavoro.

Potete anche modificare le dimensioni del testo, rendendolo più grande o più piccolo. Naturalmente, più grandi sono le dimensioni, più la vostra casella di testo dovrà essere grande per visualizzare l'intero testo. Più grandi sono le dimensioni del tipo di carattere, minore sarà la quantità di testo che la casella riuscirà a visualizzare. Per evitare di confondere l'utente più del necessario, sarebbe meglio utilizzare un'unica dimensione per tutte le caselle di testo.

Per definire le dimensioni dei caratteri di una casella di testo utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di testo della quale desiderate modificare le dimensioni del tipo di carattere.**
2. **Premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti per aprire la finestra Proprietà.**
3. **Fate doppio clic sulla proprietà Font.**

   Visual Basic visualizza una finestra di dialogo Carattere.
4. **Fate clic sulla dimensione desiderata e poi fate clic su OK.**

   Visual Basic modifica immediatamente le dimensioni del tipo di carattere del testo visualizzato nella casella prescelta.

Modificare il tipo di carattere e le dimensioni può essere talmente divertente, che Visual Basic vi offre anche altri metodi per modificare l'aspetto del testo in modo che venga visualizzato in **grassetto**, *corsivo* o ~~barrato~~. Per modificare lo stile del tipo di carattere utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di testo nella quale volete modificare il tipo di carattere.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate doppio clic sulla proprietà Font.**

   Visual Basic visualizza una finestra di dialogo Carattere.
4. **Fate clic sullo stile desiderato e poi fate clic su OK.**

   Visual Basic modifica immediatamente lo stile del tipo di carattere nella casella di testo prescelta.

Colorare le caselle di testo

Se da bambini amavate disegnare con le matite colorate, apprezzerete molto l'idea di colorare le caselle di testo utilizzando Visual Basic.

Mettete alla prova le vostre nuove conoscenze

1. **Indicate due possibili utilizzi per le caselle di testo.**

   a. Per memorizzare le lettere del gioco Scarabeo e per conservare le parole in grado di farvi vincere milioni alla Ruota della Fortuna.

   b. Per visualizzare sullo schermo il testo e permettere agli utenti di digitare un testo in un programma.

   c. Per memorizzare tutti i libri di informatica che acquistate e non leggerete mai e per costruire fortini di cartone dentro i quali i vostri figli possono nascondersi.

   d. Per utilizzarle come lettiera e dare al vostro gatto qualcosa da leggere.

2. **Se la proprietà PasswordChar di una casella di testo è impostata a * (asterisco) e la proprietà MaxLength è impostata a 10, che cosa succede?**

   a. Devo riguardare un paio di pagine del libro per trovare la risposta, quindi dovete aspettare un attimo.

   b. Non ne sono certo, ma deve trattarsi di qualcosa di importante se la domanda è riportata qui.

   c. È possibile definire la password segreta necessaria per inserirsi nei computer del Pentagono.

   d. La casella di testo accetta un massimo di 10 caratteri e visualizza un asterisco (*) al posto dei caratteri effettivamente digitati.

Solitamente Visual Basic visualizza il testo in bianco e nero. Per aggiungere un tocco di creatività potete modificare i colori di primo piano e di sfondo delle vostre caselle di testo, come illustrato nella Figura 8.7.

Il colore all'interno della casella di testo (*colore di sfondo*) viene definito dalla proprietà BackColor. Il colore del testo (*colore di primo piano*) viene definito dalla proprietà ForeColor.

Per modificare il colore di sfondo o di primo piano di una casella di testo utilizzate la procedura di seguito riportata.

1. **Fate clic sulla casella di testo della quale desiderate modificare il colore di sfondo o di primo piano.**

2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**

3 **Fate doppio clic sulla proprietà BackColor o ForeColor e poi fate clic sulla scheda Tavolozza.**

   Visual Basic visualizza una tavolozza di colori.

Figura 8.7 I colori di primo piano e di sfondo di una casella di testo

4. **Fate clic sul colore desiderato.**

   Visual Basic modifica immediatamente il colore della casella di testo prescelta.

Modificando il colore delle caselle di testo è possibile evidenziare determinate informazioni e renderle più (o meno) evidenti per l'utente. Ricordate però che troppi colori possono essere motivo di distrazione e che certe persone sono daltoniche e questo significa che non potranno apprezzare appieno il vostro programma se non potranno leggere il testo visualizzato in particolari colori. I colori devono essere utilizzati con parsimonia.

Capitolo 9

Barre di scorrimento ed etichette

In questo capitolo

- Creare le barre di scorrimento
- Spostare la casella di scorrimento
- Creare le etichette
- Allineare e mandare a capo il testo

Non tutte le scelte della vita possono essere suddivise in categorie ben definite ed elencate in oggetti come le caselle di controllo, i pulsanti di opzione e le caselle di riepilogo. Talvolta gli utenti dovranno prendere decisioni che richiedono un'ampia gamma di adattamenti graduali.

Pensate alla regolazione del volume su uno stereo. Se le uniche scelte disponibili fossero basso, medio e alto, non potreste regolare il volume come volete. Ecco perché la maggior parte degli impianti stereo vi permette di girare una manopola oppure di premere un pulsante che regola gradualmente il volume alzandolo o abbassandolo.

Per effettuare misurazioni precise oppure per visualizzare lunghi elenchi di informazioni utilizzate le barre di scorrimento. Sebbene le caselle di testo, le form e le caselle di riepilogo posseggano delle barre di scorrimento incorporate, potete creare anche delle barre di scorrimento separate. Gli elementi di una barra di scorrimento sono illustrati nella Figura 9.1.

Il CD-ROM comprende un programma in Visual Basic che illustra il funzionamento della barra di scorrimento verticale. Analizzate il codice sorgente e modificatelo per scoprire cosa succede.

Creare le barre di scorrimento

Visual Basic vi permette di creare due tipi di barre di scorrimento: orizzontali e verticali.

Figura 9.1 Anatomia di una barra di scorrimento dissezionata per una vista migliore

Le barre di scorrimento orizzontali si spostano a sinistra e a destra, proprio come i pulsanti indietro e avanti veloce del vostro videoregistratore o lettore di nastri. Le barre di scorrimento verticali si spostano in alto e in basso, proprio come il controllo del volume su alcuni stereo.

Per creare una barra di scorrimento utilizzate la procedura di seguito riportata.

1. **Fate clic sull'icona Horizontal o Vertical Scroll Bar nella casella degli strumenti di Visual Basic.**
2. **Spostate il cursore del mouse sul punto della form in cui volete disegnare la barra di scorrimento.**
3. **Tenete premuto il pulsante sinistro del mouse e muovete il mouse per disegnare la barra di scorrimento; poi rilasciate il pulsante sinistro del mouse.**
4. **Ripetete le operazioni dalla 1 alla 3 fino a quando avrete disegnato tutte le barre di scorrimento che volete creare.**

Se fate doppio clic sull'icona Horizontal o Vertical Scroll Bar nella casella degli strumenti di Visual Basic il programma crea all'istante una barra di scorrimento sulla form.

Impostare i valori minimi e massimi di una barra di scorrimento

Le barre di scorrimento sono in realtà delle rappresentazioni grafiche di valori numerici, che possono andare da -32.768 a 32.767 e possono rappresentare qualunque cosa, come misure o quantità.

Per default, Visual Basic imposta il valore massimo a 32.767 e il valore minimo a 0. Sulle barre di scorrimento orizzontali il valore massimo viene raggiunto quando la

casella di scorrimento occupa la posizione sull'estrema destra della barra; il valore minimo viene invece raggiunto quando la casella di scorrimento si trova all'estrema sinistra.

Sulle barre verticali il valore massimo si ottiene quando la casella di scorrimento si trova all'estremità inferiore della barra, mentre il valore minimo viene raggiunto quando la casella di scorrimento si trova nella posizione più alta della barra.

Ovviamente, i valori di default di 32.767 e 0 possono essere eccessivi per la maggior parte dei programmi. Per definire una gamma minore di valori, dovete modificare i valori Max e Min della barra di scorrimento.

Per modificare tali valori utilizzate la procedura di seguito riportata.

1. **Fate clic sulla barra di scorrimento della quale desiderate modificare i valori Max e Min.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate doppio clic (oppure fate clic una volta) sulla proprietà Max e digitate un nuovo valore.**
4. **Fate doppio clic (oppure fate clic una volta) sulla proprietà Min e digitate un nuovo valore.**

Se il valore Min è superiore al valore Max, la barra di scorrimento funzionerà alla rovescia. In questo caso la casella di scorrimento raggiungerà il valore massimo quando occuperà la posizione più a sinistra o più in alto e il valore minimo quando occuperà la posizione più a destra o più in basso.

Dove appare la casella di scorrimento sulle mie barre?

Per default, Visual Basic attribuisce alle barre di scorrimento il valore definito dalla proprietà Tlin (come 0); ciò significa che se i vostri valori Max e Min sono positivi, la casella di scorrimento apparirà sempre nella posizione più alta nel caso di una barra di scorrimento verticale e nella posizione più a sinistra nel caso di una barra di scorrimento orizzontale, come illustrato nella Figura 9.2.

Per default, la casella di scorrimento indica il valore minimo possibile, ma se desiderate che le vostre barre di scorrimento visualizzino un valore di default diverso dal valore minimo, potete modificare la proprietà Value della barra di scorrimento.

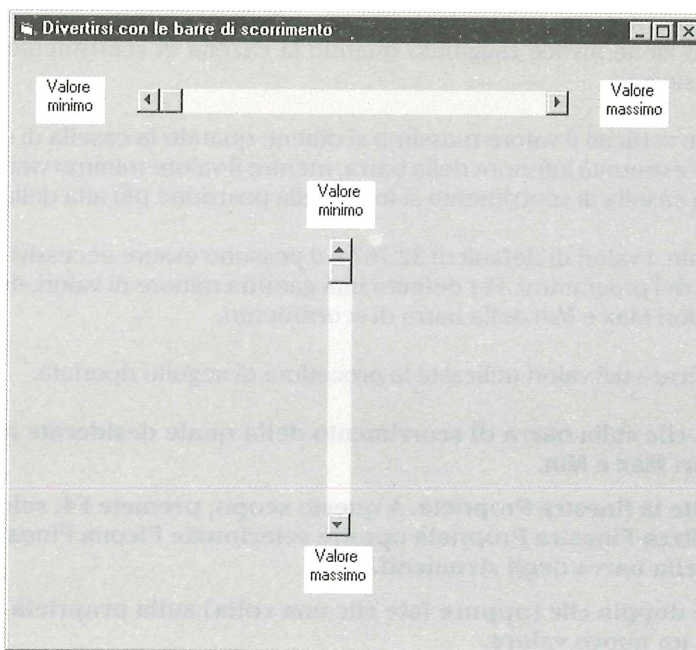



Figura 9.2 La posizione di default delle caselle di scorrimento

Per modificare tale valore utilizzate la procedura di seguito riportata.

1. **Fate clic sulla barra di scorrimento della quale desiderate modificare il valore.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate clic sulla proprietà Value e digitate un nuovo valore.**

   Visual Basic modifica all'istante il valore della barra di scorrimento.

Spostare la casella di scorrimento

La casella di scorrimento rappresenta il valore attuale della barra di scorrimento. Per spostare la casella di scorrimento gli utenti possono eseguire una delle seguenti operazioni:

- ✓ trascinare la casella di scorrimento all'interno della barra di scorrimento;
- ✓ fare clic sulle frecce di scorrimento poste all'estremità della barra di scorrimento;

✓ fare clic nell'area compresa tra la casella di scorrimento e ciascuna freccia di scorrimento.

Ogni volta che l'utente fa clic sulle frecce di scorrimento la casella di scorrimento si sposta di una determinata distanza, che per default è pari a 1. Di conseguenza, se il valore Min è 0 e il valore Max è 12, dovete fare clic 12 volte sulle frecce per spostare la casella di scorrimento da un'estremità della barra all'altra.

Per modificare la distanza percorsa dalla casella ogni volta che l'utente fa clic sulla freccia utilizzate la procedura di seguito riportata.

1. **Fate clic sulla barra di scorrimento che volete modificare.**
2. **Premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti per aprire la finestra Proprietà.**
3. **Fate doppio clic (oppure fate clic una volta) sulla proprietà SmallChange e digitate un nuovo valore.**

In modo analogo, ogniqualvolta l'utente fa clic sul campo della barra di scorrimento (ovvero la parte di barra non occupata dalla casella e dalle frecce di scorrimento), la casella di scorrimento si sposta di una determinata distanza, che per

Mettete alla prova le vostre nuove conoscenze

1. Quando si utilizza una barra di scorrimento?

a. Quando si vuole dare l'illusione di complessità.

b. Quando l'utente deve scegliere tra una gamma di valori possibili.

c. Quando dovete riavvolgere il rotolo di carta igienica che il gatto ha srotolato per terra.

d. Quando nient'altro sembra funzionare e voi avete esaurito le idee su come semplificare l'utilizzo del programma.

2. Quali sono i tre sistemi per spostare una casella di scorrimento all'interno di una barra di scorrimento?

a. Premere i tasti freccia, capovolgere il mouse oppure staccare l'alimentazione del computer.

b. Utilizzare la telecinesi, le minacce verbali oppure spingere la casella di scorrimento con le dita.

c. Non si muove mai. Era una domanda trabocchetto, vero?

d. Trascinare la casella di scorrimento con il mouse, fare clic sulle frecce di scorrimento oppure fare clic all'interno della barra di scorrimento.

default è pari a 1; ciò significa che se il valore Min è 0 e il valore Max è 5, dovrete fare clic cinque volte per spostare la casella di scorrimento da un'estremità della barra all'altra.

Per modificare la distanza coperta dalla casella quando l'utente fa clic sul campo di scorrimento utilizzate la procedura di seguito riportata.

1. **Fate clic sulla barra di scorrimento che volete modificare.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate doppio clic (oppure fate clic una volta) sulla proprietà LargeChange e digitate un nuovo valore.**

I valori delle proprietà SmallChange e LargeChange possono variare da 1 a 32.767; più piccolo è il valore, minore sarà lo spostamento della casella. Maggiore sarà il valore, più grande sarà la distanza percorsa dalla casella.

Ecco un'informazione che potreste anche non utilizzare mai. È possibile assegnare alle proprietà LargeChange e SmallChange valori maggiori del valore Max. Questo significa semplicemente che quando l'utente fa clic sulla barra di scorrimento, la casella di scorrimento si sposta immediatamente a un'estremità della barra di scorrimento.

Creare le etichette

A puro scopo decorativo, potete spargere etichette su tutte le vostre form Visual Basic. Le etichette si limitano a identificare gli oggetti presenti sulla form. Nella vita reale si incontrano etichette dovunque, per esempio con l'indicazione UOMINI o DONNE sulla porta di una toilette oppure ESTINTORE in un edificio pubblico. Le etichette si limitano ad attirare la vostra attenzione su qualcosa che altrimenti potreste non notare.

Sia le caselle di testo (vedi Capitolo 8) sia le etichette possono visualizzare un testo sullo schermo. La differenza principale sta nel fatto che l'utente può modificare il testo all'interno di una casella di testo, ma non può modificare il testo in un'etichetta.

Sebbene gli utenti non possano modificare il testo di un'etichetta, il codice BASIC è in grado di farlo. In questo modo, le vostre etichette possono essere utilizzate per inviare all'utente messaggi di vario tipo, come: "Spiacente, questa opzione al momento non è disponibile", "Pagina 2 di 9 in fase di stampa" oppure "Sei forse stupido?". Nella Figura 9.3 sono riportati alcuni esempi di etichette.

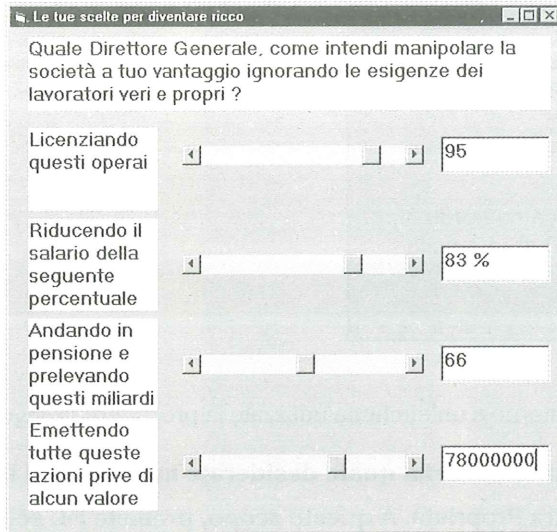



Figura 9.3 Esempi di etichette in un programma reale

Per creare un'etichetta utilizzate la procedura di seguito riportata.

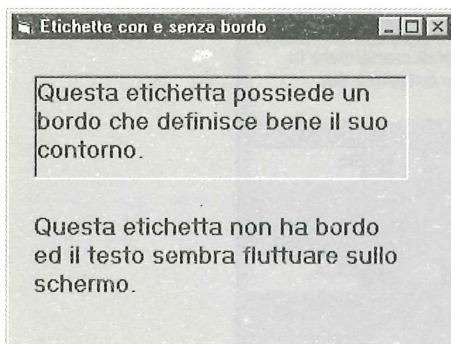
1. **Fate clic sull'icona Label nella casella degli strumenti di Visual Basic.**
2. **Spostate il cursore del mouse sul punto della form in cui volete disegnare l'etichetta.**
3. **Tenete premuto il pulsante sinistro del mouse e muovete il mouse per disegnare l'etichetta; poi rilasciate il pulsante sinistro del mouse.**

   Visual Basic disegna un'etichetta con un titolo banale, come Label2.
4. **Ripetete le operazioni dalla 1 alla 3 fino a quando avrete disegnato tutte le etichette che volete creare.**

Se fate doppio clic sull'icona Label nella casella degli strumenti di Visual Basic il programma crea all'istante un'etichetta sulla form.

Aggiungere graziosi bordi attorno alle etichette

Le etichette di solito non hanno bordi, ma se volete potete evidenziarle con un profilo per renderle più visibili. Visual Basic vi offre due scelte per i bordi delle etichette: una linea singola fissa o niente del tutto, come illustrato nella Figura 9.4.

Figura 9.4 Un'etichetta con bordo e una senza bordo

Per creare un bordo attorno a un'etichetta utilizzate la procedura di seguito riportata.

1. **Fate clic sull'etichetta della quale desiderate modificare il bordo.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate clic sulla proprietà BorderStyle e selezionate una delle seguenti opzioni:**
   - 0 (None);
   - 1 (Fixed Single).

Modificare le dimensioni delle etichette

Le dimensioni di un'etichetta sullo schermo determinano la lunghezza del titolo che la stessa etichetta può contenere; se l'etichetta è troppo piccola, una parte del titolo risulterà tagliata.

Notate che il titolo dell'etichetta (il testo) non modifica le proprie dimensioni al variare delle dimensioni dell'etichetta; per modificare le dimensioni del testo visualizzato dovete agire sulla proprietà Font.

Dato che potreste non sapere di quanto spazio necessitano i vostri titoli, continuare a modificare la larghezza e l'altezza delle vostre etichette potrebbe essere una vera tortura. Visto che i computer sono bravi a fare le cose che le persone non vogliono fare, lasciate che sia Visual Basic a occuparsi di questi dettagli.

Visual Basic è in grado di regolare automaticamente le dimensioni di un'etichetta per adattarla al testo; le etichette con adattamento automatico sono perfette per visualizzare messaggi con lunghezza variabile. Nella Figura 9.5 viene illustrata la differenza tra un'etichetta con adattamento automatico e una priva di questa caratteristica.

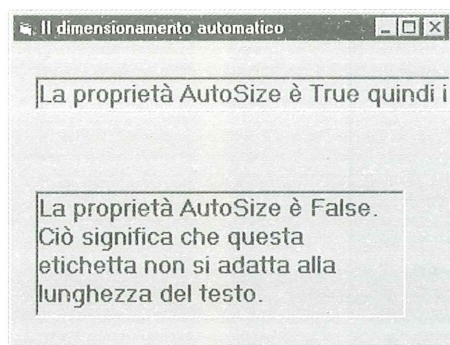



Figura 9.5 Un'etichetta con adattamento automatico e un'etichetta normale

Un'etichetta con adattamento automatico si espande o si restringe orizzontalmente per adattarsi alla lunghezza del titolo. Quindi, se avete un titolo particolarmente lungo, l'etichetta si allungherà e scomparirà oltre il lato destro dello schermo. Per fare in modo che l'etichetta con adattamento automatico mandi automaticamente a capo il testo dovete impostare la proprietà WordWrap dell'etichetta a True.

Per creare un'etichetta con adattamento automatico utilizzate la procedura di seguito riportata.

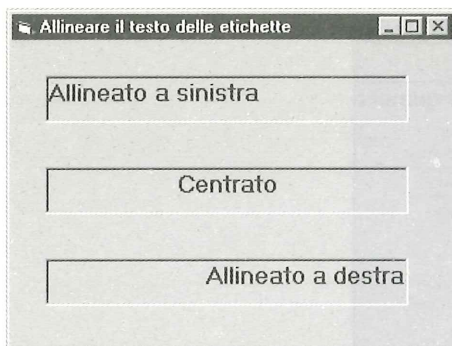
1. **Fate clic sull'etichetta che volete si regoli automaticamente adattandosi alle dimensioni del titolo.**
2. **Premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti per aprire la finestra Proprietà.**
3. **Fate clic sulla proprietà AutoSize e selezionate True.**

Il vantaggio delle etichette con adattamento automatico sta nel fatto che vi permettono di utilizzare il codice BASIC per attribuire alle etichette titoli di varia lunghezza senza dovervi preoccupare delle dimensioni. Lo svantaggio sta nella mancanza di controllo sulle dimensioni massime dell'etichetta; se non state attenti, un'etichetta potrebbe diventare troppo grande e coprire gli altri elementi della vostra interfaccia utente.

Allineare il testo all'interno di un'etichetta

Per fare in modo che il testo delle etichette abbia un aspetto gradevole e organizzato, Visual Basic vi offre tre opzioni di allineamento, come illustrato nella Figura 9.6:

- ✓ allineamento a sinistra;

Figura 9.6 I tre modi per allineare i titoli in un'etichetta

- allineamento a destra;
- allineamento al centro.

Per allineare il testo di una etichetta utilizzate la procedura di seguito riportata.

1. **Fate clic sull'etichetta con il testo da allineare.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate clic sulla proprietà Alignment e selezionate una delle seguenti opzioni:**
   - 0 (Left Justify);
   - 1 (Right Justify);
   - 2 (Center).

Etichette con ritorno a capo automatico

Se impostate la proprietà AutoSize dell'etichetta a True, l'etichetta si estende orizzontalmente in modo da contenere tutto il testo inserito. Se volete che l'etichetta si estenda invece verticalmente, dovete impostare a True sia la proprietà AutoSize sia la proprietà WordWrap.

Impostare a False entrambe queste proprietà significa tagliare i titoli lunghi sul fondo se l'etichetta non è abbastanza alta, come illustrato nel primo esempio della Figura 9.7.

Impostare AutoSize a False e WordWrap a True equivale a impostare entrambe le proprietà a False, come illustrato nel secondo esempio della Figura 9.7.

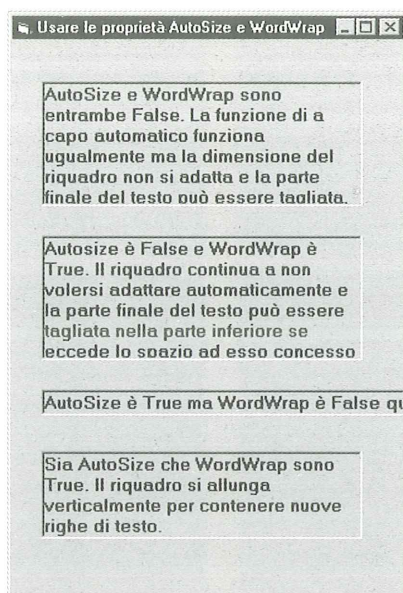



Figura 9.7 I titoli lunghi corrono il rischio di essere tagliati

Impostare AutoSize a True e WordWrap a False significa estendere l'etichetta orizzontalmente per accettare tutto il titolo, pur mostrando una sola riga di testo, come illustrato nel terzo esempio della Figura 9.7.

Impostare sia AutoSize sia WordWrap a True significa espandere l'etichetta o restringerla verticalmente, come illustrato nel quarto esempio della Figura 9.7.

Per impostare a True la proprietà WordWrap dell'etichetta utilizzate la procedura di seguito riportata.

1. **Fate clic sull'etichetta nella quale volete utilizzare il ritorno a capo automatico.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate clic sulla proprietà WordWrap e impostate il valore a True.**

Quando create delle etichette che adattano le proprie dimensioni a quella del testo, verificate che tali etichette non "crescano" troppo, tanto da coprire altri elementi dell'interfaccia utente, altrimenti confonderete davvero le idee a chiunque tenti di utilizzare il vostro programma.

Capitolo 10

Deliziosi disegni e oggetti geometrici

In questo capitolo

- Creare riquadri di disegno e riquadri di immagine
- Fare in modo che i riquadri di disegno e i riquadri di immagine si adattino al loro contenuto
- Creare forme geometriche
- Modificare l'aspetto di linee, cerchi e altre forme

È possibile spargere etichette e disegni su qualsiasi form di Visual Basic. Oltre a conferire un bell'aspetto alle vostre form, i disegni possono anche costituire parte integrante del vostro programma.

Per esempio, su una cartina stradale, un disegno che ritrae un benzinaio sorridente che dice: "Affida la tua auto all'omino con la stella" è superfluo e decorativo. Tuttavia, l'impiego su una cartina stradale di disegni che evidenziano le autostrade e le superstrade può essere fondamentale e necessario.

Se disseminate un buon numero di etichette e di disegni sulla vostra interfaccia utente, il programma risulterà più semplice da utilizzare e capire; dopotutto è proprio per questo che si creano le interfacce utente.

Creare i disegni

Visual Basic segue due modalità per visualizzare i disegni sullo schermo:

- ✓ all'interno di un riquadro di disegno;
- ✓ all'interno di un riquadro di immagine.

Un *riquadro di disegno* (picture box) viene utilizzato per visualizzare immagini grafiche o per raggruppare i pulsanti, mentre un *riquadro di immagine* (image

box) si utilizza per visualizzare immagini grafiche oppure per creare dei pulsanti grafici. Per creare un riquadro di disegno o un riquadro di immagine utilizzate la procedura di seguito riportata.

1. **Fate clic sull'icona Picture box o Image box nella casella degli strumenti di Visual Basic.**
2. **Tenete premuto il pulsante sinistro del mouse sulla form e trascinatelo nel punto in cui volete disegnare il riquadro di disegno o di immagine. Muovete il mouse per disegnare il riquadro.**
3. **Ripetete le operazioni dalla 1 alla 3 fino a quando avrete disegnato tutti i riquadri di disegno o di immagine di cui avete bisogno.**

Se fate doppio clic sull'icona Picture box o Image box nella casella degli strumenti di Visual Basic il programma disegna automaticamente il riquadro di disegno o di immagine. Dopo che avrete disegnato un riquadro di disegno o d'immagine, con ogni probabilità le dimensioni di quest'ultimo cambieranno per adattarsi alle dimensioni dell'immagine grafica che intendete collocare al suo interno.

Visualizzare dei disegni nei riquadri di disegno o di immagine

Dopo aver creato un riquadro di disegno o di immagine, la cosa più naturale è inserirvi all'interno un disegno (altrimenti perché avreste creato il riquadro?). I riquadri di disegno e di immagine possono visualizzare tre tipi di immagini grafiche.

- **File bitmap:** (che hanno estensione BMP o DIB). I *file bitmap* sono costituiti da una matrice di punti, o *pixel*, e vengono creati dai programmi di tipo paint, come Microsoft Paint. Se si ingrandisce un'immagine bitmap, questa tende ad apparire sgranata e brutta.
- **File icona:** (che hanno estensione ICO). I *file icona* sono un particolare tipo di file bitmap con dimensioni massime di 32×32 pixel.
- **Metafile:** (che hanno estensione WMF). I *metafile* sono immagini create utilizzando le linee e le forme geometriche che la maggior parte delle persone ha dimenticato appena uscita dal liceo. Questi tipi di file vengono creati dai programmi di tipo draw, come CorelDRAW.

Per caricare un disegno in un riquadro di disegno o di immagine utilizzate la procedura di seguito riportata.

1. **Fate clic sul riquadro di disegno o di immagine all'interno del quale volete caricare un file grafico.**

   Questa operazione presuppone che abbiate già disegnato il riquadro di disegno o di immagine su una form, altrimenti disegnate il riquadro di disegno o di immagine come spiegato nel paragrafo "Creare i disegni".

2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**

3. **Fate doppio clic sulla proprietà Picture nella finestra Proprietà.**

   Visual Basic visualizza la finestra di dialogo Carica immagine, come illustrato nella Figura 10.1.

4. **Fate doppio clic sul file desiderato.**

   Visual Basic carica immediatamente il disegno nel riquadro di disegno o di immagine.

Oltre a caricare i disegni utilizzando la finestra Proprietà, potete anche caricare ed eliminare le immagini mentre il programma è in esecuzione. Per caricare un'immagine nel riquadro di disegno o di immagine utilizzate il comando `LoadPicture`:

```
imgBenvenuto.Picture=
  LoadPicture("c:\graphics\martian.bmp")
```

Il comando `LoadPicture` specifica l'unità, la directory e il file da memorizzare nella proprietà Picture di un riquadro di disegno o di immagine.

Per eliminare un disegno da un riquadro di disegno o di immagine mentre il programma è in esecuzione utilizzate il comando `LoadPicture` nel seguente modo:

```
imgBenvenuto.Picture = LoadPicture("")
```

Questa istruzione carica un'immagine vuota nella proprietà Picture di un riquadro di disegno o di immagine.

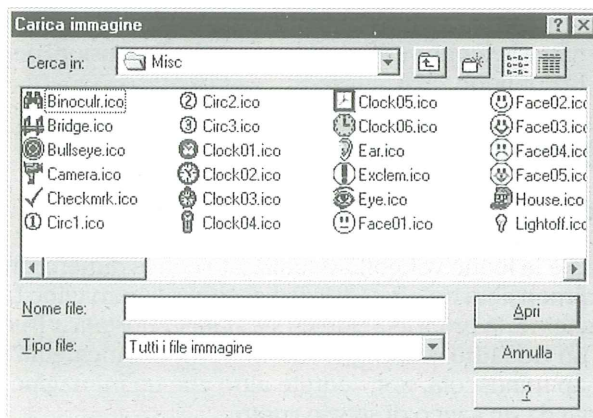



Figura 10.1 La finestra di dialogo Carica immagine

Aggiungere graziosi bordi attorno ai riquadri di disegno e di immagine

Per definire i limiti del riquadro di disegno o di immagine, Visual Basic può visualizzare un bordo. Per default, il riquadro di disegno possiede un bordo, mentre quello di immagine no. Per visualizzare o meno il bordo attorno a un riquadro di disegno o di immagine utilizzate la procedura di seguito riportata.

1. **Fate clic sul riquadro di disegno o di immagine del quale volete modificare il bordo.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate clic sulla proprietà BorderStyle e selezionate una delle seguenti opzioni:**
   - 0 None;
   - 1 Fixed Single.

Modificare le dimensioni dei riquadri di disegno o dei riquadri di immagine

Di solito le dimensioni dei riquadri di disegno o di immagine non hanno alcun effetto sulle dimensioni dell'immagine grafica visualizzata, tranne due eccezioni:

- ✓ le immagini grafiche di tipo metafile modificano sempre le proprie dimensioni per adattarsi a un riquadro di disegno o di immagine;
- ✓ se la proprietà Stretch di un riquadro di immagine viene impostata a True, le immagini bitmap e le icone varieranno le proprie dimensioni per adattarsi al riquadro che le contiene.

Modificare le dimensioni delle immagini grafiche

Le immagini bitmap e le icone vengono visualizzate con le dimensioni originali, a prescindere dalle dimensioni del riquadro di disegno o di immagine (a meno che, in un riquadro immagine, la proprietà Stretch sia stata impostata a True). Di conseguenza, se create un riquadro di disegno enorme ma vi caricate all'interno solo un'immagine bitmap minuscola, non vedrete altro che un'immagine bitmap piccolissima circondata da un sacco di spazio vuoto.

A differenza delle immagini bitmap e delle icone, i metafile si espandono o si restringono in modo da riempire tutto il riquadro di disegno oppure modificano le dimensioni del riquadro di immagine in modo da farle combaciare con le loro dimensioni originali. Per modificare le dimensioni di un metafile contenuto in un riquadro di disegno è sufficiente ridimensionare il riquadro.

Se utilizzate un riquadro di disegno, non potrete mai (ma proprio mai) modificare le dimensioni di una bitmap o di un'icona. Se invece utilizzate un riquadro di immagine, potete modificare le dimensioni delle bitmap o delle icone cambiando la proprietà Stretch del riquadro di immagine. Per modificare le dimensioni delle bitmap o delle icone in un riquadro di immagine utilizzate la procedura di seguito riportata.

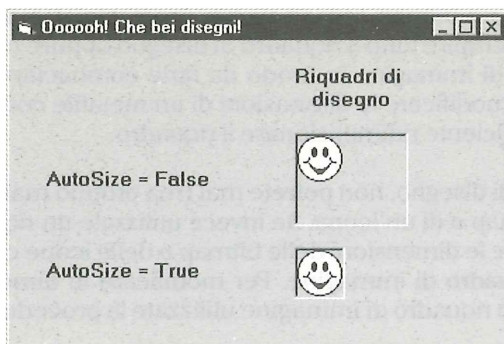
1. **Fate clic sul riquadro di immagine del quale desiderate modificare la proprietà Stretch.**
2. **Premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti per aprire la finestra Proprietà.**
3. **Fate clic sulla proprietà Stretch e impostate il valore a True.**

Quando la proprietà Stretch di un riquadro di immagine è impostata a True potete modificare le dimensioni di una bitmap o di un'icona semplicemente modificando le dimensioni del riquadro.

Modificare automaticamente le dimensioni dei riquadri di disegno

Se siete troppo occupati per preoccuparvi di creare e regolare le dimensioni dei vostri riquadri di disegno, lasciate che sia Visual Basic a farlo automaticamente per voi impostando la proprietà AutoSize a True. In questo modo, quando caricherete la bitmap o l'icona in un riquadro di disegno, quest'ultimo si ingrandirà o si rimpiccolirà immediatamente in modo da adattarsi perfettamente alla dimensione originale dell'immagine, proprio come il cellophane attorno alle scatole di dischetti (Figura 10.2). Per fare in modo che un riquadro di disegno regoli automaticamente le proprie dimensioni attorno a un'immagine grafica utilizzate la procedura di seguito riportata.

1. **Fate clic sul riquadro di disegno per il quale volete modificare la proprietà AutoSize.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate clic sulla proprietà AutoSize e impostate il valore a True.**

Figura 10.2 Confronto tra i riquadri di disegno con la proprietà AutoSize impostata a True o False

Colorare i riquadri di disegno

Visual Basic utilizza di solito un banale sfondo grigio per i riquadri di disegno. Se le vostre immagini grafiche occupano tutto il riquadro di disegno, il colore dello sfondo non ha alcuna importanza, ma se l'immagine non è sufficientemente grande, il colore dello sfondo risulterà visibile.

Il colore dello sfondo viene definito dalla proprietà BackColor. Utilizzando un colore di sfondo gradevole, riuscirete a dare maggiore risalto alle vostre immagini rendendole anche più vivaci.

Mettete alla prova le vostre nuove conoscenze

1. Quale tipo di riquadro vi permette di modificare le dimensioni di una bitmap o di un'icona: il riquadro di disegno o il riquadro di immagine?

a. Nessuno dei due. Le bitmap e le icone sono già perfette così e sarebbe assurdo pensare di poterle migliorare.

b. Un riquadro di disegno, dato che non ho la più pallida idea di che differenza ci sia.

c. Un riquadro di immagine, ma soltanto se si imposta a True la proprietà Stretch.

d. Questa domanda è troppo difficile. Ho bisogno di un sonnellino.

2. Quali sono i tre tipi di file di grafica che potete caricare in un riquadro di disegno o di immagine?

a. Bitmap, icone e metafile.

b. Bitmap, graffiti e riproduzioni di quadri famosi.

c. Bitmap, manifesti e home video.

d. Bigfoot, UFO e alcuni ritratti di vostra suocera.

Per modificare il colore di sfondo di un riquadro di disegno utilizzate la procedura di seguito riportata.

1. **Fate clic sul riquadro di disegno del quale volete modificare il colore dello sfondo.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate doppio clic sulla proprietà BackColor nella finestra Proprietà e poi fate clic sulla scheda Tavolozza.**
4. **Fate clic sul colore prescelto.**

   Visual Basic esegue immediatamente.

Linee, cerchi e altri incubi geometrici

Qual è quella cosa che possiede solo una funzione decorativa e nulla di più? Se avete risposto "il vice presidente degli Stati Uniti" ci siete andati vicini, ma la risposta corretta è: gli elementi di un'interfaccia utente che servono ad abbellirla.

Se qualcosa ha un bell'aspetto, la gente si sente ben disposta nei suoi confronti, il che spiega perché le persone fisicamente attraenti hanno vita più facile. Quindi, se riuscirete ad abbellire la vostra interfaccia utente, è probabile che molte più persone vorranno provare a utilizzarla.

Visual Basic offre sette oggetti che possono essere utilizzati per il "makeup" della vostra interfaccia utente, come illustrato nella Figura 10.3.

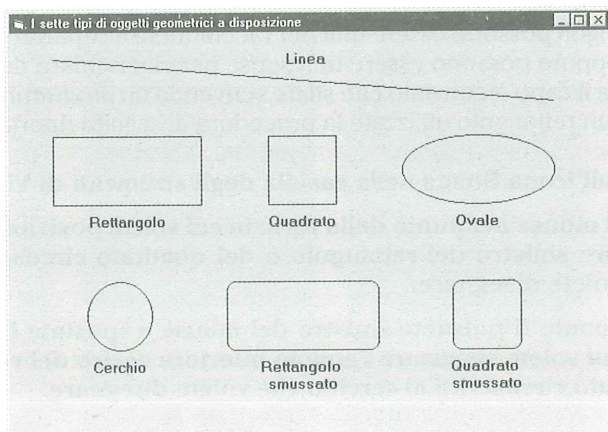



Figura 10.3 Gli oggetti geometrici che potreste inserire sulla vostra interfaccia utente

Questi oggetti sono:

- ✓ linee;
- ✓ quadrati;
- ✓ rettangoli;
- ✓ ovali;
- ✓ cerchi;
- ✓ rettangoli arrotondati;
- ✓ quadrati arrotondati.

Creare le linee

Le linee sono utili per sottolineare o separare gli elementi presenti sullo schermo. Per creare una linea utilizzate la procedura riportata alla pagina successiva.

1. **Fate clic sull'icona Line nella casella degli strumenti di Visual Basic.**
2. **Spostate il mouse nel punto della form in cui volete posizionare l'inizio della linea.**
3. **Tenete premuto il pulsante del mouse e spostate il mouse nel punto in cui volete concludere la linea.**
4. **Rilasciate il pulsante del mouse.**

Se fate doppio clic sull'icona Line nella casella degli strumenti di Visual Basic il programma crea automaticamente una linea sulla form.

Creare cerchi e rettangoli

I cerchi e i rettangoli possono essere utili per racchiudere e separare gli elementi sullo schermo oppure possono essere una scusa per cincischiare durante le ore di lavoro, mentre il capo è convinto che stiate scrivendo un programma. Per creare un cerchio o un rettangolo utilizzate la procedura di seguito riportata.

1. **Fate clic sull'icona Shape nella casella degli strumenti di Visual Basic.**
2. **Spostate il mouse nel punto della form in cui volete posizionare l'angolo superiore sinistro del rettangolo o del quadrato circoscritti al cerchio che volete disegnare.**
3. **Tenete premuto il pulsante sinistro del mouse e spostate il mouse nel punto in cui volete sistemare l'angolo inferiore destro del rettangolo o del quadrato circoscritti al cerchio che volete disegnare.**

4. **Rilasciate il pulsante sinistro del mouse.**

   A questo punto Visual Basic visualizza un rettangolo sullo schermo. Se il risultato corrisponde ai vostri desideri, non fate più nulla; in caso contrario passate al punto 5.

5. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**

6. **Fate clic sulla proprietà Shape e poi sulla freccia nella casella delle impostazioni.**

   Visual Basic visualizza un elenco delle forme disponibili:

   0 Rectangle;

   1 Square;

   2 Oval;

   3 Circle;

   4 Rounded Rectangle;

   5 Rounded Square.

7. **Fate clic sulla forma che volete creare.**

Modificare il colore delle linee e delle altre form

Visual Basic disegna di solito le linee, i cerchi e i rettangoli utilizzando una linea nera continua. Sebbene quel banale nero vada benissimo per la maggior parte degli scopi, talvolta un tocco di colore può ravvivare un'interfaccia utente. Il colore di una riga viene definito dalla proprietà BorderColor. Per modificare il colore utilizzato per una linea o un'altra forma, utilizzate la procedura di seguito riportata.

1. **Fate clic sulla linea o sulla forma della quale desiderate modificare il colore.**

2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**

3. **Fate doppio clic sulla proprietà BorderColor nella finestra Proprietà e poi fate clic sulla scheda Tavolozza.**

4. **Fate clic sul colore desiderato.**
   Visual Basic obbedisce immediatamente.

Modificare lo spessore delle linee

Le linee possono avere uno spessore variabile da 1 a 8.192 (i valori sono relativi e non corrispondono a una scala di misura reale). Una linea con uno spessore superiore a 100 tende tuttavia ad assomigliare a un salsicciotto. Per modificare lo spessore di una linea utilizzate la procedura di seguito riportata.

1. **Fate clic sulla linea o sulla forma della quale volete modificare lo spessore.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate doppio clic sulla proprietà BorderWidth e digitate un nuovo valore.**

   Visual Basic modifica immediatamente lo spessore della linea.
4. **Rimanete a bocca aperta di fronte alle meraviglie della tecnologia. Un giorno potrete dire ai vostri nipoti: "Quando andavo a scuola, dovevo disegnare le righe utilizzando il righello e la squadra. Voi ragazzi non fate più nessuna fatica con i computer e tutto il resto".**

Modificare l'aspetto di linee, cerchi e rettangoli

Visual Basic disegna di solito le linee, i cerchi e i rettangoli utilizzando una linea continua. Sebbene una linea continua sia più visibile, potreste voler creare degli effetti speciali simili a perforazioni o al codice Morse. Visual Basic offre i seguenti sette stili, alcuni dei quali sono illustrati nella Figura 10.4.

- ✓ Transparent
- ✓ Solid (il valore di default)
- ✓ Dash
- ✓ Dot
- ✓ Dash-Dot
- ✓ Dash-Dot-Dot
- ✓ Inside Solid.

Se lo spessore della linea è superiore a 1, le uniche impostazioni possibili per la proprietà BorderStyle saranno 1 (Solid) o 6 (Inside Solid). Se utilizzate un valore diverso per BorderStyle non succede nulla e potreste pensare che Visual Basic si sia guastato.

I differenti stili di linea

Stile della linea

Solid

Dash

Dot

Dash-Dot

Dash-Dot-Dot

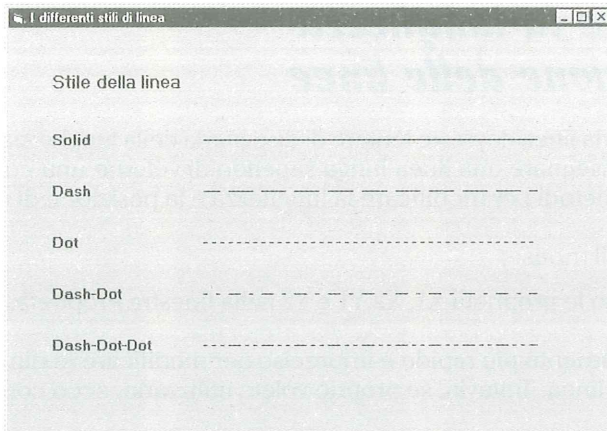



Figura 10.4 Gli esempi di alcuni stili di linea disponibili

Tanto per confondere ulteriormente le idee, l'aspetto di una linea è determinato dalla proprietà BorderStyle; solitamente un bordo è qualcosa che circonda un oggetto, mentre secondo la logica contorta di Visual Basic, BorderStyle definisce l'aspetto di una riga.

Per modificare l'aspetto delle righe o delle linee che formano i cerchi e i rettangoli utilizzate la procedura di seguito riportata.

1. **Fate clic sulla riga, sul cerchio o sul rettangolo del quale volete modificare l'aspetto.**
2. **Premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti per aprire la finestra Proprietà.**
3. **Fate clic sulla proprietà BorderStyle e selezionate una delle seguenti opzioni:**

   0 Transparent

   1 Solid

   2 Dash

   3 Dot

   4 Dash-Dot

   5 Dash-Dot-Dot

   6 Inside Solid.

Se scegliete una delle opzioni da 2 a 5, impostate la proprietà BorderWidth a 1; in caso contrario, Visual Basic utilizzerà comunque lo stile 1 (Solid).

Modificare la lunghezza e la posizione delle linee

Quando create una linea dovreste tentare di disegnarla della lunghezza esatta (che senso avrebbe disegnare una linea lunga sapendo di volerne una corta?). Visual Basic offre due metodi per modificare la lunghezza e la posizione di una linea:

- ✓ utilizzando il mouse;
- ✓ modificando le proprietà X1, X2,Y1 e Y2 nella finestra Proprietà.

Il mouse è lo strumento più rapido e impreciso per modificare le dimensioni e la posizione di una linea. Tuttavia, se proprio volete utilizzarlo, ecco come fare.

1. **Fate clic sulla linea che volete modificare.**

   Visual Basic visualizza un rettangolo blu su ognuna delle due estremità della linea; tali rettangoli vengono definiti maniglie. Dato che fare clic su una linea può essere davvero frustrante, per selezionarla potete anche spostare il mouse sopra la linea, tenere premuto il pulsante del mouse e spostare il mouse sotto la linea, rilasciando quindi il pulsante del mouse.

2. **Spostate il mouse su una di queste maniglie fino a quando il puntatore si trasforma in una croce.**
3. **Tenete premuto il pulsante del mouse e trascinate il mouse per modificare la linea. Quando la linea avrà acquisito la forma desiderata rilasciate il pulsante del mouse.**

Se preferite non utilizzare il mouse, potete utilizzare un metodo più ricercato, prediletto dalle persone raffinate e distinte: la finestra Proprietà. Per modificare le dimensioni di una linea con la finestra Proprietà utilizzate la procedura di seguito riportata.

1. **Fate clic sulla linea che volete modificare.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate doppio clic sulla proprietà X1 e digitate un nuovo valore.**
4. **Fate doppio clic sulla proprietà Y1 e digitate un nuovo valore.**
5. **Fate doppio clic sulla proprietà X2 e digitate un nuovo valore.**
6. **Fate doppio clic sulla proprietà Y2 e digitate un nuovo valore.**

   Nella Figura 10.5 sono riportate le coordinate x e y di una linea.

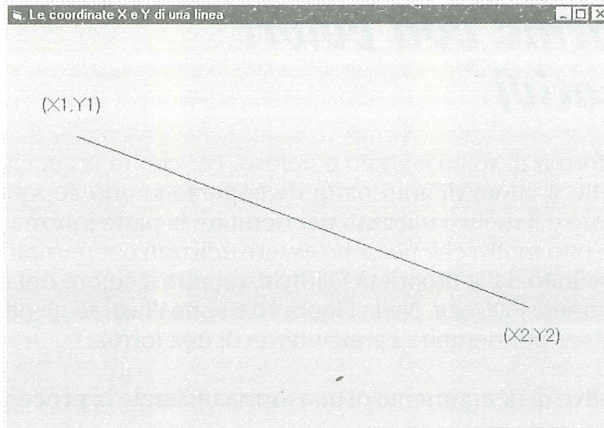



Figura 10.5 In che modo le coordinate X1, X2, Y1 e Y2 influiscono sulla posizione di una linea

Modificare le dimensioni di cerchi, rettangoli e altre forme

Per fortuna, modificare le dimensioni di cerchi, rettangoli e altre forme è molto più semplice che modificare le dimensioni di una linea. A questo scopo, potete utilizzare il mouse o la finestra Proprietà. Per modificare le dimensioni di una forma con il mouse utilizzate la procedura di seguito riportata.

1. **Fate clic sulla forma che volete modificare.**

   Visual Basic visualizza delle maniglie nere attorno alla forma.

2. **Spostate il mouse sopra una delle maniglie fino a quando il puntatore si trasforma in una freccia a due punte.**
3. **Tenete premuto il pulsante del mouse e spostate il mouse; quando l'oggetto assume la forma desiderata, rilasciate il pulsante del mouse.**

Coloro che preferiscono utilizzare la tastiera a scapito della semplicità e della comodità, possono modificare le dimensioni di una forma utilizzando la finestra Proprietà. A tale scopo utilizzate la procedura di seguito riportata.

1. **Fate clic sulla forma che volete modificare.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate doppio clic sulla proprietà Height e digitate un nuovo valore.**
4. **Fate doppio clic sulla proprietà Width e digitate un nuovo valore.**

Riempire le forme con colori e splendidi motivi

L'interno di una forma di solito è vuoto e noioso. Per creare un'eccitazione maggiore di quella che il cuore di gran parte delle persone può sopportare, potete modificare il colore e il motivo utilizzati per riempire la parte interna delle forme. Visual Basic offre otto motivi che possono essere utilizzati per riempire una forma. Il motivo viene definito dalla proprietà FillStyle, mentre il colore del motivo viene definito dalla proprietà FillColor. Nella Figura 10.6 sono illustrati gli otto motivi che Visual Basic fornisce per riempire l'area interna di una forma.

Per definire il motivo di riempimento di una forma utilizzate la procedura di seguito riportata.

1. **Fate clic sulla forma che volete riempire.**
2. **Premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti per aprire la finestra Proprietà.**
3. **Fate clic sulla proprietà FillStyle e selezionate una delle seguenti opzioni:**

   0 Solid

   1 Transparent

   2 Horizontal Line

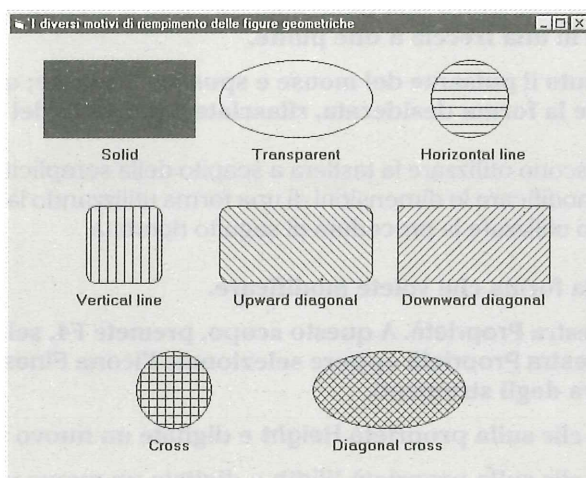



Figura 10.6 Gli otto motivi di riempimento disponibili

3 Vertical Line

4 Upward Diagonal

5 Downward Diagonal

6 Cross

7 Diagonal Cross.

Per modificare il colore di un motivo di riempimento utilizzate la procedura di seguito riportata.

1. **Fate clic sulla forma della quale volete modificare il colore di riempimento.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate doppio clic sulla proprietà FillColor nella finestra Proprietà e poi fate clic sulla scheda Tavolozza.**
4. **Fate clic sul colore desiderato.**

   Visual Basic obbedisce immediatamente.

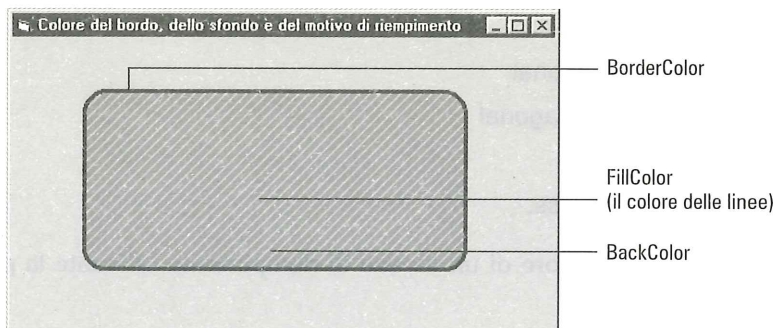
Modificare il colore di sfondo delle forme

Oltre a poter modificare il colore del motivo di riempimento all'interno delle forme (FillColor) e il colore utilizzato per disegnare la forma (BorderColor), potete anche modificare il colore di sfondo delle forme (BackColor). Siete confusi? Date un'occhiata alla Figura 10.7 e tutto vi apparirà più chiaro.

Prima di poter modificare il colore di sfondo di una forma dovete impostare la proprietà BackStyle di tale forma a Opaque (il valore di default è Transparent, ovvero trasparente; se lo sfondo è trasparente, non ha alcun senso modificarne il colore).

Per modificare il colore di sfondo di una forma utilizzate la procedura di seguito riportata.

1. **Fate clic sulla forma della quale volete modificare il colore di sfondo.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure selezionate l'icona Finestra proprietà nella barra degli strumenti.**
3. **Fate clic sulla proprietà BackStyle e impostate il valore e Opaque.**

Figura 10.7 Il colore di sfondo di una forma rispetto al bordo e al colore di riempimento

4. **Fate doppio clic sulla proprietà BackColor nella finestra Proprietà e poi fate clic sulla scheda Tavolozza.**
5. **Fate clic sul colore desiderato.**

   Visual Basic obbedisce immediatamente.

Provate anche voi a modificare le dimensioni di un oggetto

Il programma d'esempio riportato di seguito vi permette di modificare lo spessore di un cerchio utilizzando la barra di scorrimento orizzontale. Per sperimentare direttamente quando appena detto create tre oggetti con i seguenti valori di proprietà. Se non avete voglia di creare il programma presentato di seguito, caricate ed eseguite il file SHAPE.VBP incluso sul CD.

Oggetto	*Proprietà*	*Valore*
Form	Caption	Il cerchio che si allarga e si restringe
Forma1	(Name)	shpCerchio
	Height	1455
	Left	2640
	Shape	3 (Circle)
	Top	1080
Width	1695	

(continua)

Oggetto	*Proprietà*	*Valore*
BarraScorOr1	(Name)	hsbCerchio
	Height	255
	Left	720
	Max	20
	Min	1
	Top	360
	Width	3255

Fate doppio clic sulla barra di scorrimento orizzontale e digitate quanto segue nella finestra Codice:

```
Private Sub hsbCerchio_Change()
  shpCerchio.BorderWidth = hsbCerchio.Value
End Sub
```

Per eseguire il programma premete F5 poi fate clic sulla barra di scorrimento orizzontale e osservate il cerchio crescere davanti ai vostri occhi. Sorprendente! Lascerete i vostri amici a bocca aperta!

Parte III

Creare i menu

In questa parte...

I menu a discesa sono eccezionali per organizzare tutte le opzioni offerte dal programma. In questo modo, se gli utenti vogliono svolgere delle operazioni con il vostro programma, devono semplicemente scegliere il menu a discesa corretto e selezionare il comando appropriato.

In questa parte del libro viene spiegato come creare i menu a discesa nei vostri programmi. Liberi di crederci o meno, creare menu personalizzati è veramente semplice. La parte difficile è far funzionare effettivamente il programma nel modo desiderato, compito che neppure i programmatori di Microsoft riescono sempre a svolgere.

Capitolo 11

Creare e modificare i menu a discesa

In questo capitolo

- Creare i menu e le voci dei menu
- Aggiungere le barre di separazione
- Utilizzare i tasti di scelta rapida e i segni di spunta
- Disattivare o far scomparire i comandi dei menu

In genere, ogni programma comprende i seguenti menu: File, Modifica, Finestra e ? (Guida), come illustrato nella Figura 11.1. Il menu File compare all'estrema sinistra, seguito dal menu Modifica, il penultimo è il menu Finestra e infine, sull'estrema destra, si trova il menu ? (Guida); tra i menu Modifica e Finestra si inseriscono normalmente i menu specifici del programma.

Ogni menu non comprende solo i titoli di menu, ma anche delle voci, come illustrato nella Figura 11.2. I titoli compaiono nella parte superiore dello schermo in una barra dei menu, mentre le voci che corrispondono ai vari comandi vengono visualizzate solo quando viene selezionato il menu al quale appartengono.

Gli elementi di base di una barra dei menu

Prima di creare dei menu dovete decidere quanti utilizzarne e sotto quale menu inserire i vari comandi.

Figura 11.1 Un tipico esempio di menu a discesa

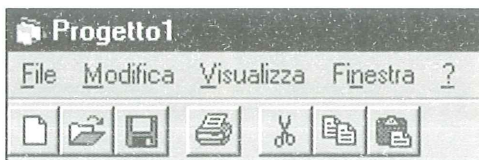
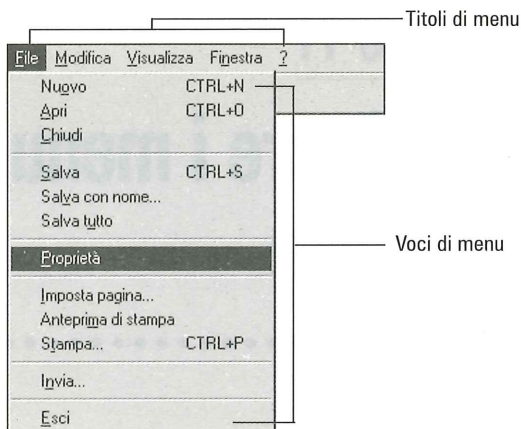



Figura 11.2 I titoli di menu e le voci di menu

Visual Basic è in grado di creare i menu automaticamente se utilizzate la funzione Creazione guidata applicazioni VB per creare il vostro programma (come illustrato nel Capitolo 3).

Il menu File (Figura 11.2) dovrebbe contenere i comandi direttamente correlati alle operazioni relative ai file, quali l'apertura, la chiusura, il salvataggio e la stampa dei file, nonché l'uscita da un programma, in modo che possiate andare in cucina a farvi uno spuntino.

Il menu Modifica, illustrato nella Figura 11.3. dovrebbe contenere i comandi correlati alla modifica, quali Annulla (e Ripeti), Taglia, Copia, Incolla, Cancella e Seleziona tutto.

Il menu Finestra, illustrato nella Figura 11.4, dovrebbe contenere i comandi correlati all'apertura, alla chiusura, alla disposizione e al passaggio tra diverse finestre.

Il menu ? (Guida), illustrato nella Figura 11.5, dovrebbe contenere i comandi utili per ottenere un aiuto dal programma. Solitamente, i comandi della guida comprendono un sommario del sistema di guida, un indice alfabetico, pubblicità sull'assistenza e un inutile comando Informazioni su..., che visualizza informazioni che secondo i programmatori fanno un figurone sullo schermo.

Modifica
Annulla CTRL+Z
Taglia CTRL+X
Copia CTRL+C
Incolla CTRL+V
Incolla speciale...

Figura 11.3 Un tipico menu Modifica

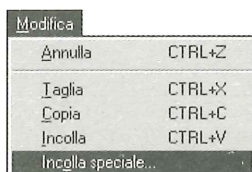



Figura 11.4 Un tipico menu Finestra

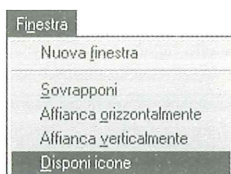
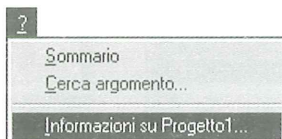



Figura 11.5 Un tipico menu ? (Guida)

Qualunque altro menu inserito tra i menu Modifica e Finestra dovrebbe classificare in modo chiaro il tipo di comandi contenuti.

Per esempio, molti programmi di elaborazione testi contengono il menu Strumenti che visualizza i comandi per il controllo ortografico, la suddivisione sillabica, la creazione di macro e altri ancora che il 99% degli utenti non utilizzerà mai.

Se i titoli dei vostri menu sono esclusivi del vostro programma (ovvero se non sono i soliti titoli Modifica e Finestra presenti sugli altri programmi), sforzatevi di escogitare nomi descritti, in modo che gli utenti possono intuire dove cercare un determinato comando.

Creare dei menu per la vostra interfaccia utente

Per creare e modificare i menu dovete aprire la finestra Editor di menu, illustrata nella Figura 11.6. Se utilizzate la funzione Creazione guidata applicazioni VB , Visual Basic è in grado di creare automaticamente menu a discesa standard, ma dovrete comunque utilizzare la finestra Editor di menu per modificare i menu.

In nome della libertà e della confusione Visual Basic offre tre metodi per visualizzare la finestra Editor di menu:

- ✓ premere CTRL+E;
- ✓ selezionare Strumenti-Editor di menu...;
- ✓ fare clic sull'icona Editor di menu nella barra degli strumenti (Figura 11.6).

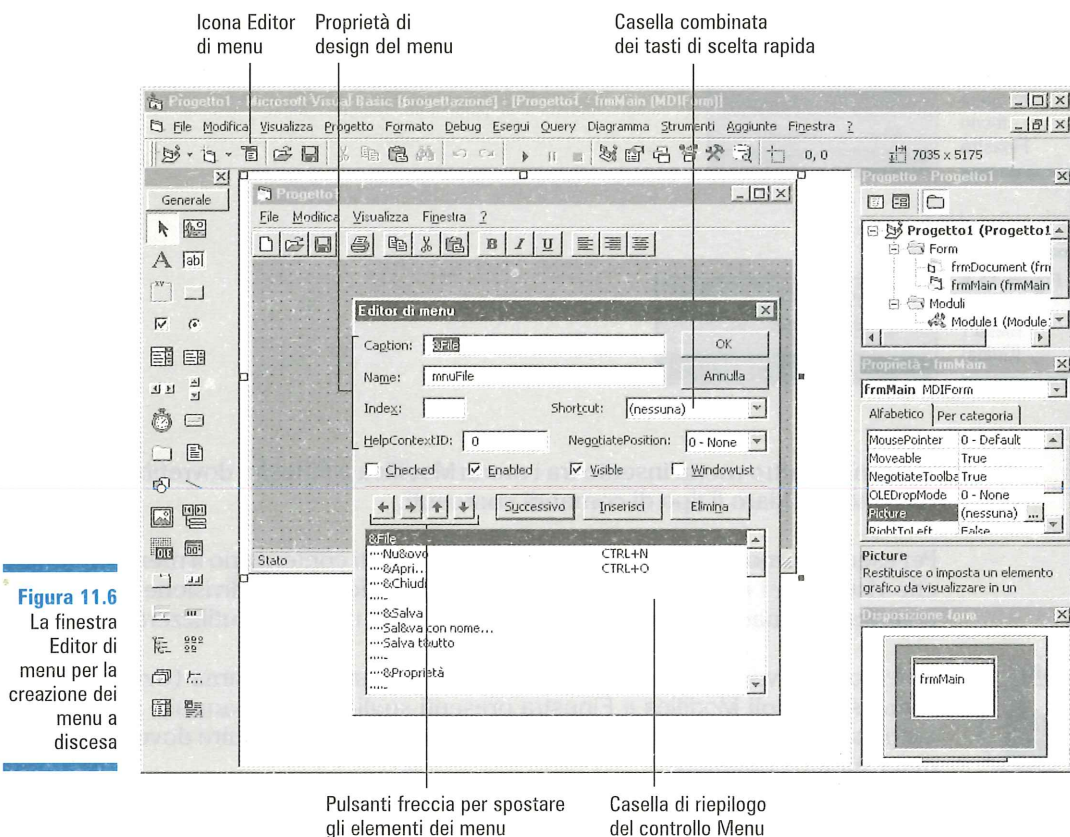



Figura 11.6 La finestra Editor di menu per la creazione dei menu a discesa

È possibile creare una serie di menu per ciascuna form, quindi se il programma ne contiene due, potrete disporre di un menu completamente diverso per ciascuna di esse. Ovviamente questa soluzione potrebbe confondere gli utenti, ma se siete dei tipici programmatori, che di solito non si curano di ciò che pensano gli utenti, questo problema non vi preoccuperà più di tanto.

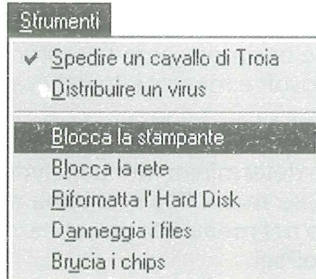
La finestra Editor di menu è il luogo dove è possibile definire ogni dettaglio per creare i menu; i primi due elementi da definire per tutti i vostri menu sono i nomi e i relativi titoli.

Attribuire un nome ai menu

Ogni menu e voce di menu dispone di un titolo e di un nome. Il titolo è ciò che compare sullo schermo, mentre il nome non compare mai sullo schermo, ma serve per identificare quale menu o voce di menu viene scelta dall'utente.

I titoli possono comprendere un massimo di 40 caratteri, compresi numeri, spazi, simboli di punteggiatura e il carattere di sottolineatura (_). Ovviamente, più il titolo è lungo, maggiore sarà lo spazio occupato sullo schermo.

Dato che i titoli compaiono sullo schermo, potete utilizzare la "e" commerciale (&) per creare titoli come &File oppure T&abella. Qual è lo scopo di una cosa tanto stupida? La "e" commerciale posta di fronte a una qualsiasi lettera fa in modo che tale lettera del titolo venga sottolineata, come illustrato nella Figura 11.7.

Figura 11.7 La "&" inserita in un nome visualizza tramite sottolineatura le scelte rapide dalla tastiera

Quando il titolo di un menu presenta una lettera sottolineata gli utenti possono visualizzare quel menu premendo ALT insieme alla lettera sottolineata. Se, per esempio, un titolo di menu viene denominato &Finestra, tale titolo viene visualizzato sullo schermo come Finestra e gli utenti possono visualizzare il menu premendo ALT+F. Questa possibilità può aiutare gli utenti a selezionare i titoli servendosi delle scelte rapide mediante tastiera al posto del mouse.

Quando la "e" commerciale sottolinea una lettera del titolo di una voce di menu, gli utenti possono selezionare tale voce digitando la lettera sottolineata, senza premere ALT. Quindi, se per esempio una voce di menu viene denominata &Nuovo, essa viene visualizzata come Nuovo e gli utenti possono selezionarla aprendo prima il menu e poi premendo N.

I nomi, come i titoli, possono utilizzare un massimo di 40 caratteri, compresi i numeri e il carattere di sottolineatura. Dato che i nomi non compaiono mai sullo schermo è possibile digitare nomi lunghi, fino a raggiungere la lunghezza massima di 40 caratteri.

Contrariamente ai titoli, i nomi non possono comprendere spazi, simboli di punteggiatura o parolacce.

Per quanto riguarda i nomi dei menu, Visual Basic consiglia di iniziare con mnu, come nei seguenti esempi:

- mnuFile

- ✓ mnuFinestra
- ✓ mnuFileApri

Visual Basic non si preoccupa dell'utilizzo coerente dei caratteri maiuscoli e minuscoli. Volendo, avreste potuto scrivere i nomi anche così:

- ✓ MNufiLemNufINEstra
- ✓ MNUfileAPRI

Questi nomi non solo sono difficili da leggere, ma vi fanno anche sembrare degli analfabeti. Quindi, per amor di coerenza e per proteggere la vostra immagine, il metodo migliore è adottare lo stile di Microsoft e mantenerlo ogni volta che utilizzate Visual Basic.

Per identificare le voci che appaiono sotto determinati menu includete il titolo di menu nel nome della voce. Per esempio, se menu File si chiama mnuFile, i comandi Apri, Salva ed Esci che compaiono nel menu File potrebbero avere nomi quali mnuFileApri, mnuFileSalva e mnuFileEsci.

Creare i menu

La creazione dei menu a discesa per un programma Visual Basic comporta l'esecuzione di una procedura in due fasi:

- ✓ innanzitutto occorre creare i menu che appariranno sulla barra dei menu;
- ✓ poi occorre creare le voci dei menu che compariranno sotto ciascun menu.

Per creare i menu che dovranno comparire nella barra dei menu in cima alla form utilizzate la procedura di seguito riportata.

1. **Fate clic sulla form alla quale volete aggiungere dei titoli di menu.**
2. **Premete CTRL+E, selezionate Strumenti-Editor di menu oppure fate clic sull'icona Editor di menu nella barra degli strumenti per aprire la finestra Editor di menu.**
3. **Nella casella di testo Caption della finestra Editor di menu, digitate il titolo del menu che volete far apparire sullo schermo, comprese le "e" commerciali (&).**

   Nel frattempo, Visual Basic visualizza il titolo nella casella di riepilogo del controllo Menu.
4. **Premete TAB per spostare il cursore nella casella di testo Name.**
5. **Digitate il nome del menu cominciando con mnu e proseguendo con il titolo del menu stesso, come mnuFile o mnuModifica.**

Potete utilizzare indifferentemente caratteri maiuscoli e minuscoli, ma per coerenza con i programmatori Visual Basic di tutto il mondo, rimanete fedeli allo stile mnuFileEsci e utilizzare le maiuscole solo per distinguere le singole parole come File ed Esci.

6. **Premete INVIO oppure fate clic su Next per creare il successivo titolo di menu.**

7. **Ripetete le operazioni dalla 3 alla 6 fino ad aver creato tutti i titoli di menu che dovranno comparire in cima allo schermo nella barra dei menu.**

8. **Fate clic su OK.**

   Visual Basic visualizza i menu nella parte superiore della form.

Aggiungere e cancellare menu e voci di menu

La creazione di menu è piuttosto semplice. Purtroppo, nella vita nulla è eterno, nemmeno i menu. Prima o poi vi capiterà di dover aggiungere o cancellare menu.

Per aggiungere un nuovo menu a una form utilizzate la procedura di seguito riportata.

1. **Fate clic sulla form alla quale volete aggiungere un nuovo menu.**

2. **Aprite la finestra Editor di menu. A questo scopo, premete CTRL+E, selezionate Strumenti-Editor di menu oppure fate clic sull'icona Editor di menu nella barra degli strumenti.**

3. **Fate clic sul titolo di menu che volete far comparire a destra rispetto al vostro nuovo menu.**

4. **Fate clic su Inserisci.**

   Visual Basic spinge in basso il titolo di menu evidenziato in precedenza e inserisce una riga vuota, come illustrato nella Figura 10.8.

5. **Fate clic sulla casella di testo Caption e digitate il nuovo titolo di menu, come &Strumenti o Forma&to.**

6. **Premete TAB per spostare il cursore sulla casella di testo Name e digitate il nuovo nome del menu (come mnStrumenti o mnuFormato) e poi premete INVIO.**

7. **Fate clic su OK.**

Per cancellare un titolo di menu utilizzate la procedura di seguito riportata.

1. **Fate clic sulla form dalla quale volete cancellare un titolo di menu.**

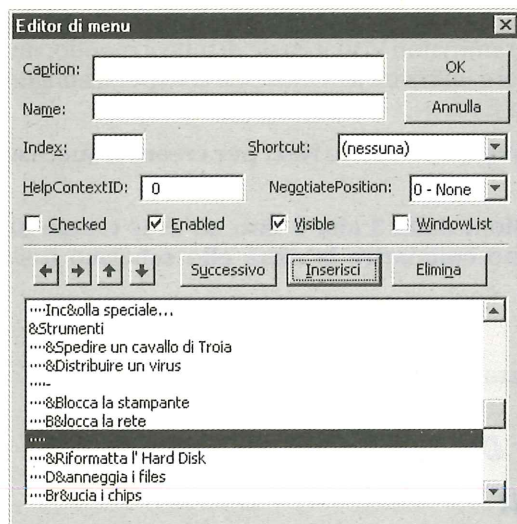



Figura 11.8 L'inserimento di un nuovo titolo di menu nella casella di riepilogo del controllo Menu

2. **Aprite la finestra Editor di menu. A questo scopo, premete CTRL+E, selezionate Strumenti-Editor di menu oppure fate clic sull'icona Editor di menu nella barra degli strumenti.**

3. **Fate clic sul titolo di menu che volete cancellare.**

4. **Fate clic su Elimina.**

   Visual Basic cancella il titolo di menu evidenziato.

5. **Fate clic su OK.**

Quando cancellate un titolo di menu o un comando di menu, tutto il codice BASIC scritto per detto comando continuerà a esistere, per cui dovrete cancellare anche questo.

Aggiungere delle voci ai menu

Dopo aver creato i menu che compariranno nella barra dei menu in cima alla form, la fase successiva consiste nel creare le voci che dovranno comparire sotto ciascuno di essi.

Nella casella di riepilogo controllo Menu, tutte le voci allineate a sinistra sono titoli di menu che compaiono nella barra dei menu, mentre le voci rientrate sono voci di menu che compaiono sotto il titolo di menu.

Per creare le voci dei menu utilizzate la procedura di seguito riportata.

1. **Fate clic sulla form alla quale volete aggiungere delle voci di menu.**
2. **Premete CTRL+E, selezionate Strumenti-Editor di menu oppure fate clic sull'icona Editor di menu nella barra degli strumenti per aprire la finestra Editor di menu.**

   Visual Basic apre all'istante la finestra Editor di menu.
3. **Fate clic sotto il titolo del menu all'interno del quale volete inserire la nuova voce.**

   Per esempio, se volete inserire una voce all'interno del menu File, fate clic sotto al relativo titolo.
4. **Fate clic su Inserisci.**
5. **Fate clic nella casella di testo Caption e digitate il titolo della voce di menu, come &Salva o Sta&mpa.**
6. **Premete TAB per spostare il cursore nella casella di testo Name.**
7. **Digitate il nome associato alla voce di menu, come mnuFileSalva o mnuFileStampa.**
8. **Fate clic sul pulsante con la freccia rivolta a destra per far rientrare la voce di menu.**

   Tale rientro indica che si tratta di una voce di menu e non di un titolo di menu (che ne dite di tutti questi termini simili che confondono decisamente il tutto?).
9. **Fate clic su OK.**

Spostare i titoli e le voci dei menu

Quando create i vostri menu a discesa potete sempre modificarli in un momento successivo. Visual Basic offre quattro possibilità per spostare i titoli e le voci dei menu:

- ✓ in alto
- ✓ in basso
- ✓ rientrato a destra
- ✓ rientrato a sinistra.

Nella finestra Editor di menu Visual Basic fornisce quattro pulsanti freccia che vi permettono di spostare le voci in alto, in basso, a destra o a sinistra (Figura 11.6). Lo spostamento di una voce in alto o in basso nella casella di riepilogo del controllo Menu serve a riorganizzare la posizione di un elemento sulla barra dei menu o nel menu a discesa.

Per spostare un elemento in alto o in basso nella casella di riepilogo del controllo Menu utilizzate la procedura di seguito riportata.

1. **Fate clic sulla form che contiene i titoli o le voci di menu che volete spostare.**
2. **Aprite la finestra Editor di menu. A questo scopo, premete CTRL+E, selezionate Strumenti-Editor di menu oppure fate clic sull'icona Editor di menu nella barra degli strumenti.**
3. **Fate clic sull'elemento che volete spostare in alto o in basso nella finestra Editor di menu.**
4. **Fate clic sul pulsante con la freccia rivolta verso l'alto per spostare l'elemento verso l'alto oppure sul pulsante con la freccia rivolta verso il basso per spostare l'elemento verso il basso.**
5. **Fate clic sul pulsante OK quando avete finito di trastullarvi.**

All'interno della finestra Editor di menu, facendo rientrare un titolo a destra lo si trasforma in una voce di menu (Figura 11.9). In modo analogo, facendo rientrare una voce a sinistra la si trasforma in un titolo di menu.

Per far rientrare un elemento a destra o a sinistra nella casella di riepilogo del controllo Menu utilizzate la procedura di seguito riportata.

1. **Fate clic sulla form che contiene i titoli o le voci di menu che volete spostare.**

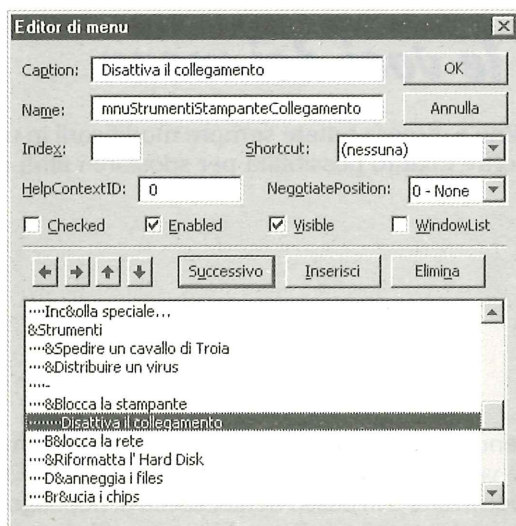



Figura 11.9 L'effetto del rientro verso destra di una voce

2. **Aprite la finestra Editor di menu. A questo scopo, premete CTRL+E, selezionate Strumenti-Editor di menu oppure fate clic sull'icona Editor di menu nella barra degli strumenti.**

3. **Fate clic sull'elemento che volete spostare a destra o a sinistra nella finestra Editor di menu.**

4. **Fate clic sul pulsante con la freccia rivolta a destra per far rientrare l'elemento verso destra oppure sul pulsante con la freccia a sinistra per far rientrare l'elemento verso sinistra.**

5. **Fate clic sul pulsante OK.**

Abbellire i menu

I menu a discesa elencano i comandi in modo tale che gli utenti possano trovarli (almeno si spera). Per rendere i menu ancora più semplici da utilizzare Visual Basic vi permette anche di separare i comandi di menu con barre di separazione, di visualizzare i segni di spunta accanto alle voci di menu selezionate, di aggiungere i tasti di scelta rapida per velocizzare l'accesso ai comandi più utilizzati, di disattivare o cancellare determinate voci.

Inserire delle barre di separazione

Le *barre di separazione* sono righe inserite in un menu e utilizzate per dividere gruppi di voci, come illustrato nella Figura 11.10. In genere, le barre di separazione raggruppano tre o più voci correlate in modo tale che gli utenti riescano rapidamente a individuare il comando desiderato.

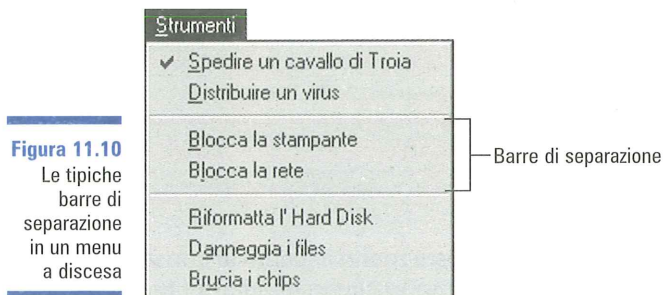



Figura 11.10 Le tipiche barre di separazione in un menu a discesa

Per creare una barra di separazione utilizzate la procedura di seguito riportata.

1. **Fate clic sulla form che contiene i menu ai quali volete aggiungere le barre di separazione.**

2. **Aprite la finestra Editor di menu. A questo scopo, premete CTRL+E, selezionate Strumenti-Editor di menu oppure fate clic sull'icona Editor di menu nella barra degli strumenti.**

3. **Fate clic sulla voce nella finestra Editor di menu che volete far apparire subito sotto la barra di separazione.**

4. **Fate clic su Inserisci in modo che Visual Basic visualizzi una riga vuota.**

   Se necessario, fate clic sui pulsanti, → ← in modo che la barra di separazione compaia sullo stesso livello delle voci che state dividendo.

5. **Fate clic nella casella di testo Caption, digitate un trattino (-) e poi premete TAB per spostare il cursore nella casella di testo Name.**

6. **Digitate il nome che identifichi la barra di separazione.**

   Di solito il nome dovrebbe comprendere il titolo del menu, come mnuFileBar o mnuEditBar3.

7. **Fate clic su OK per chiudere la finestra Editor di menu.**

Assegnare i tasti di scelta rapida

Dopo un po', essere costretti a utilizzare continuamente i menu a discesa per selezionare un comando può diventare noioso. Nel caso dei comandi utilizzati più di frequente sarebbe meglio assegnare dei tasti di scelta rapida, come CTRL+S per selezionare il comando Salva oppure CTRL+X per il comando Taglia. I tasti di scelta rapida permettono all'utente di selezionare un comando senza dover passare dai menu a discesa; nella Figura 11.11 sono illustrati alcuni tasti di scelta rapida.

Figura 11.11 Esempi di tasti di scelta rapida in un menu a discesa

I tasti di scelta rapida vengono visualizzati sui menu accanto ai comandi che rappresentano; in questo modo gli utenti potranno capire rapidamente quali tasti di scelta rapida vengono offerti per tutti i comandi dei menu.

Per assegnare un tasto di scelta rapida a un comando di menu dovete utilizzare nuovamente la finestra Editor di menu. Anche se a volte potrebbe essere gratificante inventare dei tasti di scelta rapida personalizzati, Visual Basic vi permette di scegliere solo da un limitato elenco di tasti possibili. Visual Basic non vi permette

Mettete alla prova le vostre nuove conoscenze

1. **Perché i menu a discesa sono così utili?**

   a. Nascondono i comandi impedendo agli utenti di trovarli.

   b. Rendono i programmi più semplici da utilizzare organizzando i comandi correlati in menu facili da trovare.

   c. I menu a discesa sono utili per confondere gli utenti e allo stesso tempo dare loro l'illusione che sia colpa loro per il fatto di non sapere come si utilizza il programma.

   d. Se i menu a discesa sono così utili, perché la gente continua ad acquistare manuali di 400 pagine per imparare a utilizzare questi stupidi programmi?

2. **Come potete creare e modificare i menu a discesa in un programma Visual Basic?**

   a. Premendo CTRL+E, seleziondo Strumenti-Editor di menu oppure facendo clic sull'icona Editor di menu nella casella degli strumenti per aprire l'Editor di menu.

   b. Copiando il programma di qualcun altro nella speranza che nessuno se ne accorga.

   c. Creare menu a discesa senza ricorrere alla scrittura di programmi complessi in C++ e Java è praticamente impossibile.

   d. Visual Basic è in grado di creare menu a discesa?

di assegnare gli stessi tasti di scelta rapida a comandi diversi. Nel caso doveste provarci, Visual Basic vi rimprovererà visualizzando un messaggio di errore.

Per assegnare i tasti di scelta rapida ai comandi dei menu utilizzate la procedura di seguito riportata.

1. **Fate clic sulla form che contiene i menu ai quali volete aggiungere i tasti di scelta rapida.**
2. **Aprite la finestra Editor di menu. A questo scopo, premete CTRL+E, selezionate Strumenti-Editor di menu oppure fate clic sull'icona Editor di menu nella barra degli strumenti.**

   Visual Basic visualizza la finestra Editor di menu.
3. **Fate clic sul comando del menu al quale volete assegnare un tasto di scelta rapida.**
4. **Fate clic sul pulsante ↓ nella casella di riepilogo Shortcut.**

   Visual Basic visualizza un elenco di possibili combinazioni di tasti da utilizzare.
5. **Fate scorrere l'elenco fino a quando individuate la combinazione di tasti corretta.**

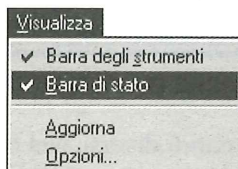
In teoria, sarebbe meglio scegliere combinazioni di tasti facili da ricordare, come CTRL+S per il comando Salva o CTRL+X per il comando Taglia. Visual Basic visualizza la vostra scelta nella "casella di riepilogo del controllo Menu".

6. **Fate clic su OK.**

A questo punto, quando farete clic sui vostri menu a discesa i tasti di scelta rapida verranno visualizzati accanto ad alcuni comandi. Dato che non avete scritto un codice BASIC per comunicare a questi comandi che cosa fare, premendo questi tasti di scelta rapida non accadrà un bel niente.

Inserire i segni di spunta accanto alle voci di menu

I segni di spunta che compaiono accanto alle voci dei menu mostrano che la voce è già stata selezionata (Figura 11.12). I segni di spunta sono spesso utili per identificare quale tipo di carattere, stile o dimensioni sono stati selezionati.

Figura 11.12 L'utilizzo dei segni di spunta in un menu a discesa

Se volete creare delle opzioni di default nei vostri menu a discesa potete fare in modo che vengano visualizzati dei segni di spunta durante l'esecuzione del programma.

I segni di spunta possono apparire soltanto accanto alle voci di menu e non ai titoli dei menu. Se cercate di inserire un segno di spunta accanto a un titolo di menu, Visual Basic si mette a urlare e visualizza il messaggio di errore `Impossibile aggiungere segno di spunta`.

Per aggiungere un segno di spunta accanto ai comandi dei menu utilizzate la procedura di seguito riportata.

1. **Fate clic sulla form che contiene i menu ai quali volete aggiungere i segni di spunta.**
2. **Aprite la finestra Editor di menu. A questo scopo, premete CTRL+E, selezionate Strumenti-Editor di menu oppure fate clic sull'icona Editor di menu nella barra degli strumenti.**

3. **Fate clic sulla voce del menu accanto alla quale volete inserire un segno di spunta.**
4. **Fate clic sulla casella di controllo Checked.**
5. **Fate clic su OK.**

Se inserite dei segni di spunta accanto alle voci dei menu durante l'esecuzione, potreste anche doverli cancellare. A questo scopo dovete utilizzare (gasp!) il codice BASIC.

Per cancellare un segno di spunta accanto a una voce di menu, impostate a False la proprietà Checked della voce. L'esempio di seguito riportato elimina un segno di spunta da una voce di menu denominato `mnuFont12`:

```
mnuFont12.Checked = False
```

Per aggiungere un segno di spunta utilizzando il codice BASIC impostate a True la proprietà Checked della voce. L'esempio di seguito riportato aggiunge un segno di spunta accanto alla voce di menu denominata `mnuFontHelvetica`:

```
mnuFontHelvetica.Checked = True
```

Disattivare le voci di menu

Talvolta non ha senso utilizzare determinati comandi; per esempio, se non è stato selezionato un blocco di testo, è inutile disporre dei comandi Taglia o Copia. Per impedire agli utenti di selezionare comandi non disponibili potete disattivarli. In questo modo i comandi saranno visualizzati in grigio, come illustrato nella Figura 11.13, e gli utenti non potranno selezionarli.

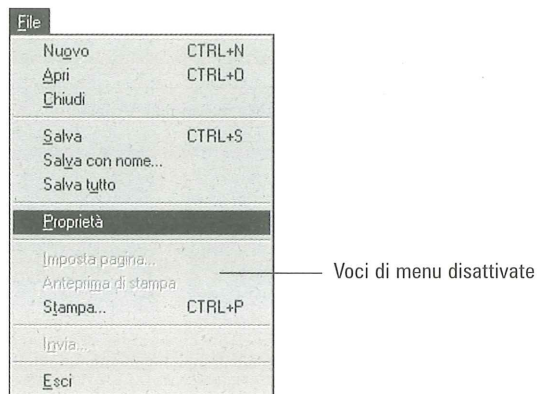



Figura 11.13 Un esempio di comandi di menu visualizzati in grigio

Per disattivare una voce di menu utilizzate la procedura di seguito riportata.

1. **Fate clic sulla form che contiene i comandi di menu che volete disattivare.**
2. **Aprite la finestra Editor di menu. A questo scopo, premete CTRL+E, selezionate Strumenti-Editor di menu oppure fate clic sull'icona Editor di menu nella barra degli strumenti.**
3. **Evidenziate la voce di menu che volete disattivare.**
4. **Fate clic sulla casella di controllo Enabled per eliminare il segno di spunta (Figura 11.14).**
5. **Fate clic su OK.**

Se disattivate una voce di menu, potreste anche volerla riattivare. A questo scopo dovete utilizzare il codice BASIC. Per riattivare una voce di menu impostate a True la proprietà Enabled. L'esempio di seguito riportato riattiva una voce denominata `mnuModificaTaglia`:

```
mnuModificaTaglia.Enabled = True
```

Per disattivare una voce di menu mentre il programma è in esecuzione utilizzate il codice BASIC e impostate a False la proprietà Enabled della voce. L'esempio di seguito riportato disattiva la voce `mnuModificaCopia`:

```
mnuModificaCopia.Enabled = False.
```

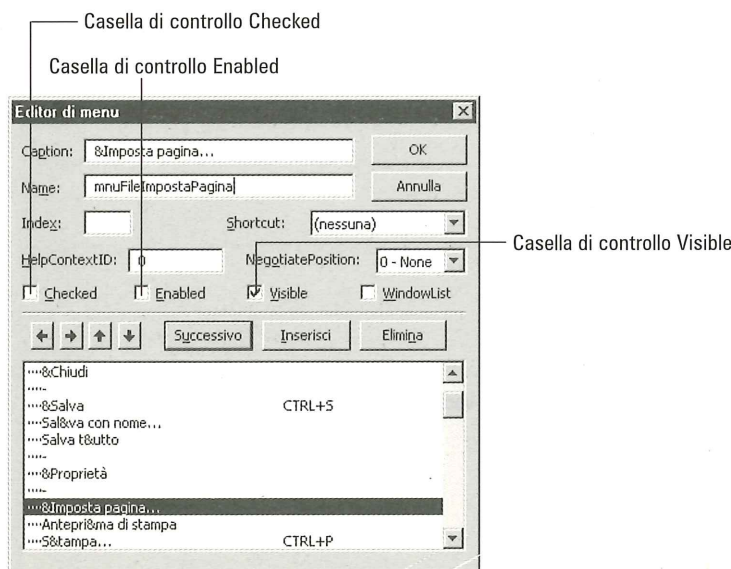



Figura 11.14 La finestra Editor di menu

Far scomparire le voci di menu

Invece di disattivare una voce, potete anche farla scomparire. Per esempio, alcuni programmi eliminano tutti i titoli di menu, lasciando soltanto File e ? (Guida), finché l'utente non ha aperto o creato un file (dopotutto è inutile visualizzare un menu Modifica se non c'è nulla da modificare).

Per eliminare una voce di menu utilizzate la procedura di seguito riportata.

1. **Fate clic sulla form che contiene i comandi di menu che volete rendere invisibili.**
2. **Premete CTRL+E, selezionate Strumenti-Editor di menu oppure fate clic sull'icona Editor di menu nella barra degli strumenti per aprire la finestra editor di menu.**
3. **Evidenziate la voce di menu che volete rendere invisibile.**
4. **Fate clic sulla casella di controllo Visible per eliminare il segno di spunta (Figura 11.14).**
5. **Fate clic su OK.**

Dopo aver reso invisibile una voce potreste anche doverla rendere di nuovo visibile. A questo scopo dovete utilizzare il codice BASIC, impostando a True la proprietà Visible della voce. L'esempio di seguito riportato rende visibile il titolo di menu `mnuModifica`:

```
mnuModifica.Visible = True
```

Per rendere invisibile una voce di menu mentre il programma è in esecuzione utilizzate il codice BASIC e impostate a False la proprietà Visible della voce. L'esempio di seguito riportato rende invisibile il titolo di menu `mnuStrumenti`:

```
mnuStrumenti.Visible = False
```

Ricordate che tutti questi accorgimenti rendono il vostro programma più semplice da utilizzare e gli conferiscono un tocco di professionalità. Come potrà confermare qualunque programmatore professionista, se il vostro programma ha un bell'aspetto, quando qualcosa non funziona gli utenti daranno la colpa a se stessi. Ecco perché i programmatori impiegano un sacco di tempo per creare un'interfaccia utente: perché gli utenti non se la prendano con loro se il programma si rivela una catastrofe.

Capitolo 12

Sottomenu, menu che si espandono e menu di scelta rapida

In questo capitolo

- Creare dei sottomenu
- Menu che si espandono in modo dinamico
- Creare dei menu di scelta rapida

Una barra dei menu tipica visualizza un elenco di titoli di menu nella parte superiore dello schermo. Selezionando uno di questi viene visualizzato un menu a discesa.

Purtroppo una barra può contenere soltanto un numero limitato di menu e un menu a discesa potrebbe contenere più voci di quante ne possano apparire sullo schermo. Di conseguenza, che cosa succede se scrivete un'applicazione eccezionale che richiede più comandi di quanti ne possano essere visualizzati sulla barra dei menu o su più menu a discesa? La soluzione è l'utilizzazione dei sottomenu (oppure riscrivere il programma).

Creare dei sottomenu

I sottomenu sono spesso utilizzati per seppellire un comando sotto parecchi strati di menu a discesa. Se vengono organizzati in modo corretto, i sottomenu mostrano chiaramente la relazione tra i diversi argomenti; in caso contrario i vostri programmi non risulteranno molto diversi da quei pacchetti di poco prezzo che milioni di persone sono costrette a utilizzare tutti i giorni al lavoro.

Molti programmi dispongono per esempio di un menu Testo, all'interno del quale possono comparire voci come Stile, Carattere e Dimensioni. Quando si seleziona, per esempio, la voce Carattere, appare un sottomenu che elenca tutti i tipi di carattere disponibili, come illustrato nella Figura 12.1.

Figura 12.1 Il menu Testo offre la voce Carattere e un sottomenu con i tipi di carattere

Visual Basic vi permette di creare fino a quattro livelli di sottomenu, come illustrato nella Figura 12.2. Sebbene tutti questi livelli di sottomenu possano essere utili, la maggior parte dei programmatori utilizza un unico livello di sottomenu per evitare di relegare i comandi tanto in fondo da non consentire agli utenti di ritrovarli. Si dice che i nastri Watergate, il corpo di Jimmy Hoffa e il Sacro Graal risiedano in qualche sottomenu di un ben noto programma.

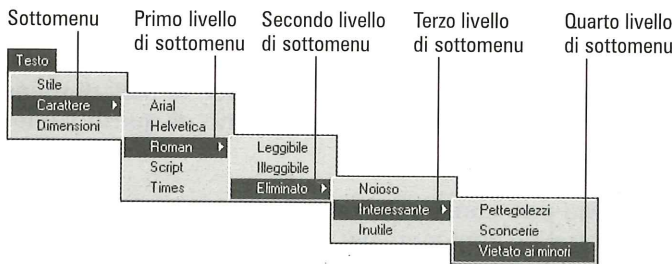
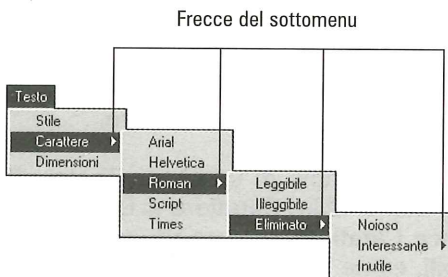
Figura 12.2 Quattro livelli di sottomenu sono più che sufficienti

Ogni volta che a lato di una voce di menu è visualizzata una freccia significa che tale voce possiede un sottomenu. Quando create dei sottomenu, Visual Basic visualizza questa freccia automaticamente; nella Figura 12.3 viene riportato un esempio di tale freccia.

Figura 12.3 La freccia accanto a un comando di menu indica l'esistenza di un sottomenu

La finestra Editor di menu è l'unico luogo in cui è possibile definire i sottomenu. Qualunque voce allineata a sinistra verrà visualizzata nella barra dei menu come titolo. Le voci con un rientro singolo appaiono invece sui menu a discesa, le voci con doppio rientro vengono visualizzate sul primo livello di sottomenu, quelle con triplo rientro sul secondo livello di sottomenu, quelle con quadruplo rientro sul terzo livello e quelle con rientro quintuplo sul quarto e ultimo livello di sottomenu.

Per creare dei sottomenu, utilizzate la procedura di seguito riportata.

1. **Fate clic sulla form sulla quale desiderate creare dei sottomenu.**
2. **Aprite la finestra Editor di menu. A questo scopo, premete CTRL+E, selezionate Strumenti-Editor di menu oppure fate clic sull'icona Editor di menu nella barra degli strumenti.**
3. **Nella casella di riepilogo del controllo Menu, evidenziate la voce di menu che volete far comparire su un sottomenu.**
4. **Fate clic sul pulsante con la freccia a destra per far rientrare la voce.**
5. **Fate clic su OK.**

Nella casella di riepilogo del controllo Menu ogni livello di rientro (livello di sottomenu) viene rappresentato da quattro puntini (....).

Se volete spostare i sottomenu trasportandoli a un livello superiore, (per esempio dal sottomenu di livello 3 al sottomenu di livello 2) potete utilizzare la procedura che segue.

1. **Fate clic sulla form che contiene i sottomenu che volete modificare.**
2. **Aprite la finestra Editor di menu. A questo scopo, premete CTRL+E, selezionate Strumenti-Editor di menu oppure fate clic sull'icona Editor di menu nella barra degli strumenti.**
3. **Nella casella di riepilogo del controllo Menu, evidenziate la voce di menu che volete spostare in alto di un livello.**
4. **Fate clic sul pulsante con la freccia a sinistra per far rientrare la voce.**
5. **Fate clic su OK.**

Invece di utilizzare molti livelli di sottomenu, la maggior parte dei programmatori migliori utilizza delle finestre di dialogo (illustrate nel Capitolo 13). Una finestra di dialogo permette agli utenti di effettuare numerose selezioni contemporaneamente e non li costringe a fare una scelta alla volta passando attraverso un'infinità di livelli di sottomenu. I sottomenu sono perfetti quando si tratta di selezionare poche opzioni; le finestre di dialogo sono la soluzione migliore quando occorre scegliere tra diverse opzioni.

Ma allora perché offrire dei sottomenu se persino Microsoft è di avviso contrario? Si tratta semplicemente di una delle possibilità offerte da Microsoft per scrivere interfacce utente difficili da utilizzare, perché in questo modo i vostri programmi non riusciranno mai a rappresentare una vera minaccia per loro.

Modificare i titoli dei menu mentre il programma è in esecuzione

In determinati casi è necessario modificare il titolo di una voce di menu mentre il programma è in esecuzione. La voce di menu che viene modificata più frequentemente è quella relativa al comando Annulla. Dopo avere selezionato il comando Annulla, la maggior parte dei programmi modifica la voce ad esso associata per visualizzare il comando Ripeti.

Per modificare il titolo di un menu, dovete utilizzare il codice BASIC; è sufficiente che troviate il nome della voce di menu che volete modificare e impostiate la sua proprietà Caption su un nuovo titolo. Nell'esempio seguente il titolo mnuModificaAnnulla viene modificato in Ripeti:

```
mnuModificaAnnulla.Caption = "Ripeti"
```

Nell'esempio seguente il titolo mnuModificaAnnulla viene modificato nuovamente in Annulla:

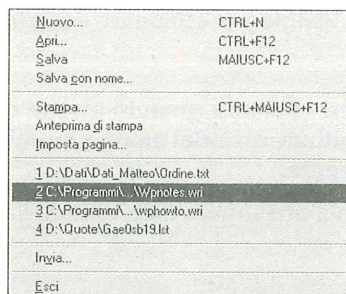
```
mnuModificaAnnulla.Caption = "Annulla"
```

Quando modificate i titoli di menu, potete utilizzare il simbolo della "e" commerciale (&) per visualizzare un comando di menu con i tasti di scelta rapida, illustrati dettagliatamente nel Capitolo 11. Il codice seguente modifica per esempio il titolo mnuModificaAnnulla in Annulla, con la A sottolineata.

```
mnuModificaAnnulla.Caption = "&Annulla"
```

Progettare menu che si espandono in modo dinamico

Se utilizzate molti programmi Windows come Microsoft Word, probabilmente noterete questa caratteristica: ogni volta che caricate un programma, il menu File visualizza l'elenco degli ultimi quattro o cinque file sui quali avete lavorato (Figura 12.4).

Figura 12.4 Un menu File a espansione dinamica nel quale sono elencati gli ultimi quattro file aperti

Se avete mai aperto due o più finestre nello stesso programma, allora forse avrete notato che anche il menu Finestra elenca i nomi dei file aperti in quel momento.

Per creare un menu che si espanda in modo dinamico, dovete innanzitutto creare degli spazi vuoti nel vostro menu. A questo scopo, utilizzate la finestra Editor di menu e create il codice BASIC necessario per aggiungere elementi in modo da renderli visibili.

Utilizzate quindi la procedura di seguito riportata.

1. **Fate clic sulla form nella quale desiderate creare un menu che si espanda in modo dinamico.**
2. **Premete CTRL+E, selezionate Strumenti-Editor di menu oppure fate clic sull'icona Editor di menu nella barra degli strumenti per aprire la finestra Editor di menu.**
3. **Fate clic sotto l'ultimo comando del menu al quale volete aggiungere nuove voci.**

   Per esempio, se volete aggiungere nuove voci sotto il menu File, fate clic sulla voce successiva all'ultimo comando del menu File.
4. **Fate clic su Inserisci per aggiungere la voce.**
5. **Fate clic sul pulsante con la freccia a destra per far rientrare la riga appena inserita, in modo che appaia come voce di menu.**
6. **Lasciate vuota la casella di testo Caption e premete TAB.**
7. **Nella casella di testo Name digitate lo stesso nome, come mnuFileMRU, per ciascuna delle righe che state inserendo.**
8. **Digitate 0 nella casella di testo Index per la prima riga vuota.**

   Ogni volta che aggiungete una riga vuota, aumentate di uno il numero della casella di testo Index; la prima riga vuota di tale casella deve contenere 0, la seconda 1 e così via.

9. **Fate clic sulla casella di controllo Visible per eliminare il segno di spunta.**
10. **Premete INVIO.**
11. **Ripetete le operazioni dalla 3 alla 10 fino a quando avrete creato quattro o cinque righe vuote sotto l'ultima voce del menu al quale desiderate aggiungere le voci dinamicamente.**

    Controllate che ogni riga vuota abbia un numero differente nella sua casella di testo Index.

12. **Fate clic su OK.**

Questa la procedura per creare un menu che si espande in modo dinamico; tuttavia, per aggiungere davvero delle voci a questo menu, dovete utilizzare il codice BASIC. Nell'esempio seguente viene illustrato come aggiungere una voce per creare un menu che si espande in modo dinamico.

```
Private Sub Form_Load()
  mnuFileMRU(0).Caption = "&1 C:\VB\HELLO.VBP"
  mnuFileMRU(0).Visible = True
End Sub
```

La seconda riga della procedura di un evento Form_Load imposta il titolo della prima riga vuota su 1 C:\VB\HELLO.VBP. La terza riga rende la voce visibile sul menu a discesa.

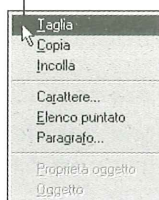
Creare dei menu di scelta rapida

I menu di scelta rapida (chiamati anche menu di contesto) vengono spesso utilizzati per visualizzare immediatamente sullo schermo un elenco di comandi. Qualunque menu o sottomenu può essere visualizzato come menu di scelta rapida. Di solito questo è programmato per apparire quando l'utente preme il pulsante destro del mouse. Nella Figura 12.5 viene illustrato un esempio di un menu di scelta rapida.

Per creare tale menu, dovete utilizzare il comando BASIC PopupMenu. Nell'esempio seguente il menu mnuModifica viene visualizzato sotto forma di menu di scelta rapida quando l'utente fa clic con il pulsante destro del mouse.

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer,
X As Single, Y As Single)
  If Button = 2 Then ' pressione tasto destro del mouse
     PopupMenu mnuModifica ' visualizza il menu Modifica
  End If
End Sub
```

Le coordinate X e Y vengono misurate dall'angolo superiore sinistro

Figura 12.5 Un esempio di un menu di scelta rapida

Invece di utilizzare il numero 1 per indicare il pulsante sinistro del mouse e il numero 2 per indicare quello destro, potete impiegare rispettivamente le variabili vbPopupMenuLeftButton e vbPopupMenuRightButton.

Per creare un menu di scelta rapida, utilizzate la procedura di seguito riportata.

1. **Fate clic sulla form che contiene il menu che volete trasformare in un menu di scelta rapida.**
2. **Selezionate Visualizza-Codice oppure fate clic sull'icona Visualizza codice nella finestra Progetto.**

   La finestra Codice viene visualizzata.
3. **Fate clic nella casella di riepilogo Oggetto e selezionate Form.**
4. **Fate clic nella casella di riepilogo Procedura e selezionate MouseDown.**

   Visual Basic visualizza una procedura vuota `Private Form_MouseDown`.
5. **Digitate il seguente codice sotto l'istruzione Private Sub Form_ MouseDown e sopra l'istruzione End Sub:**

```
If Button = vbPopupMenuRightButtom Then
  PopUpMenu (digitate il nome del menu come mnuModifica)
End If
```

Se, per qualche strana ragione, desiderate visualizzare un menu premendo il pulsante sinistro del mouse, sostituite la variabile vbPopupMenuRightButton con la variabile vbPopupMenuLeftButton.

Definire la posizione dei menu di scelta rapida

I menu di scelta rapida compaiono di solito in corrispondenza della posizione del puntatore del mouse, qualunque essa sia, ma se volete che tali menu appaiano in una posizione specifica dello schermo potete indicare delle coordinate esatte.

Per esempio, il seguente comando BASIC `PopUpMenu` visualizza un menu di scelta rapida denominato `mnuStrumenti` in corrispondenza della coordinata X `500` e della coordinata Y `650`:

```
PopUpMenu mnuStrumenti, 500, 650
```

Coloro che sono particolarmente pignoli sulla posizione del menu di scelta rapida rispetto al puntatore del mouse possono specificare se il menu debba essere visualizzato sulla sinistra, sulla destra o centrato rispetto al puntatore. A questo scopo, utilizzate i seguenti comandi:

```
PopUpMenu mnuModifica, 0 'Allineamento a sinistra
PopUpMenu mnuModifica, 4 'Allineamento al centro
PopUpMenu mnuModifica, 8 'Allineamento a destra
```

L'utilizzo di numeri per specificare l'allineamento a destra e a sinistra può essere semplice da digitare ma può anche essere fonte di confusione. Per semplificare il tutto, Visual Basic vi permette di assegnare a questi valori delle frasi perfettamente comprensibili come le seguenti:

```
PopUpMenu mnuModifica, vbPopupMenuLeftAlign "Allineato a
sinistra
PopUpMenu mnuModifica, vbPopupMenuCenterAlign "Allineato al
centro
PopUpMenu mnuModifica, vbPopupMenuRightAlign "Allineato a
destra
```

In sostanza, queste istruzioni dicono a Visual Basic: "Ehi, stupido, ogni volta che vedi le parole `vbPopUpMenuRightAlign`, sostituisci il numero 8". Utilizzando il vecchio metodo, l'aspetto di una procedura Visual Basic è il seguente:

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer,
X As Single, Y As Single)
  If Button = 2 Then ' pressione tasto destro del mouse
     PopupMenu mnuModifica, 4 ' metodo sorpassato e fuorviante
  End If
End Sub
```

Ecco invece il risultato che si ottiene grazie al sistema delle istruzioni perfettamente comprensibili:

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer,
X As Single, Y As Single)
  If Button = 2 Then ' pressione tasto destro del mouse
     PopupMenu mnuModifica, vbPopupMenuCenterAlign
  End If
End Sub
```

Nel caso abbiate già dato libero sfogo alla fantasia, ricordate che potete sempre allineare a sinistra, al centro o a destra un menu rispetto alla coordinata X di un punto specifico. Per esempio, se volete centrare un menu di scelta rispetto al punto (500, 650), utilizzate la seguente istruzione:

```
PopupMenu mnuStrumenti, 4, 500, 650
```

oppure

```
PooUpMenu mnuStrumenti, vbPopupMenuCenterAlign, 500, 650
```

In questo esempio, il menu di scelta rapida viene visualizzato sulle coordinate x e y rispettivamente di 500 e 650 twip.

Se modificate l'unità di misura passando da twip a centimetri o pollici, potrebbe essere necessario modificare i valori delle coordinate x e y per adattarli alla nuova scala. Se desiderate impostare unità di misura differenti per la vostra form, leggete il Capitolo 6.

Definire il pulsante destro del mouse in modo che funzioni con i menu di scelta rapida

I menu di scelta rapida funzionano di solito come normali menu a discesa: per selezionare un comando è sufficiente fare clic sulla voce corrispondente con il pulsante sinistro del mouse. Tuttavia, dato che il pulsante destro del mouse viene utilizzato approssimativamente con la stessa frequenza con cui utilizzate i denti del giudizio, potreste voler concedere ai vostri utenti la possibilità di utilizzarlo per selezionare le voci dei menu di scelta rapida. Nel seguente esempio di codice viene attivato il pulsante destro del mouse.

```
PopupMenu mnuModifica, 2
```

Per sostituire ai numeri una frase di senso compiuto, potete utilizzare quanto segue:

```
PopupMenu mnuModifica, vbPopupMenuRightButton
```

Per default, Visual Basic parte dal presupposto che intendiate utilizzare il pulsante sinistro del mouse con qualunque menu di scelta rapida.

Domanda trabocchetto: "Come si fa a definire contemporaneamente sia l'allineamento centrato che l'attivazione del pulsante destro del mouse?". La risposta è: "Utilizzando l'operatore OR", come illustrato nel seguente esempio:

```
PopupMenu mnuModifica, 2 Or 4
```

Mettete alla prova le vostre nuove conoscenze

1. **Spiegate per quale motivo potreste voler modificare i titoli di menu mentre il programma è in esecuzione.**

   a. Per confondere gli utenti in modo che pensino di aver commesso un errore.

   b. Per spostare comandi importanti ogni cinque minuti in modo da tenere desta l'attenzione degli utenti.

   c. Per modificare i titoli di menu come Annulla e Ripeti.

   d. Perché non si viene considerati veri programmatori se non si utilizzano tutte ma proprio tutte le funzioni offerte dal linguaggio di programmazione per rendere il programma ancora più difficile da usare.

2. **Spiegate che cosa fa il seguente codice BASIC.**

```
Private Sub
 Form_MouseDown (Button
As Integer, Shift As Integer,
X As Single, Y As Single)

If Button = vb PopupMenuRightBut-
ton Then

 PopupMenu mnuCarattere

 End If

End Sub
```

   a. Quando l'utente preme il pulsante destro del mouse, il programma visualizza il menu mnuCarattere sotto forma di menu di scelta rapida.

   b. Quando l'utente preme il pulsante destro del mouse, viene dichiarata la III° guerra mondiale.

   c. Quando l'utente preme qualsiasi pulsante, il mouse di autodistrugge con un bel Pop!

Per utilizzare termini immediatamente comprensibili al posto dei numeri, potete utilizzare quanto segue:

```
PopupMenu mnuModifica, vbPopupMenuRightButton

vbPopupMenuCenterAlign
```

La maggior parte dei programmi famosi (come WordPerfect, Excel e Paradox) utilizza sottomenu, menu che di espandono in modo dinamico e menu di scelta rapida. Tuttavia, nel caso di programmi meno complicati come i giochi, probabilmente non vi serviranno tutti questi diversi tipi di menu.

Fortunatamente la maggior parte degli utenti conosce già queste funzioni di menu e quando sullo schermo compariranno dei menu di scelta rapida o dei sottomenu, nessuno subirà uno shock. Il segreto è utilizzare queste funzioni soltanto quando è necessario. Ricordate, più elementi fantasiosi inserite nel vostro programma, più saranno le cose di cui dovrete preoccuparvi (non siete felici di aver deciso di imparare a programmare un computer?).

Capitolo 13

Finestre di dialogo

In questo capitolo

- Creare le finestre di dialogo
- Aggiungere le icone e i pulsanti alle finestre di dialogo
- Le finestre di dialogo più utilizzate

I menu a discesa rendono senza dubbio la vita più semplice agli utenti, a patto che, ovviamente, gli utenti siano in grado di utilizzarli. Oltre ai menu a discesa, quasi tutti i programmi utilizzano anche finestre di dialogo.

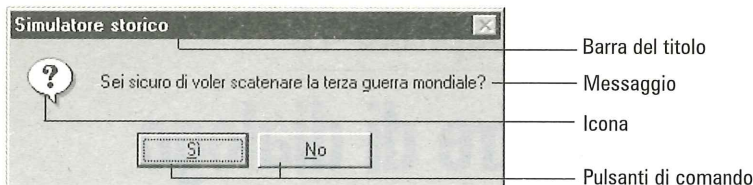
Le finestre di dialogo sono quelle piccole finestre che appaiono sullo schermo. Il computer utilizza solitamente le finestre di dialogo per far sapere all'utente che cosa sta succedendo, come "Stampa pagina 4 di 67" oppure "Windows 98 si è bloccato un'altra volta e questo è il codice di errore applicazione che non capirete".

Le finestre di dialogo permettono anche al computer di porre domande agli utenti, come "Annulla stampa?" oppure "Vuoi davvero uscire da Windows?". Una bella finestra di dialogo può essere ricca di opzioni e permettere così all'utente di compiere più scelte in un'unica volta. Proprio come la maggior parte dei programmi basati su Windows utilizza menu a discesa simili (File, Modifica, ?), anche questi programmi utilizzano finestre di dialogo simili.

Visualizzare una casella di messaggio

Una casella di messaggio non è altro che una finestra di Windows all'interno della quale vengono visualizzati, come si può vedere dalla Figura 13.1, i quattro seguenti elementi:

- ✓ la barra del titolo
- ✓ un messaggio
- ✓ un'icona in grado di attirare l'attenzione
- ✓ uno o più pulsanti di comando.

Figura 13.1 Un messaggio di errore

La barra del titolo identifica lo scopo della finestra di dialogo, per esempio Informazioni su questo programma. Il messaggio contiene il testo che appare nella finestra di dialogo, come "Sei sicuro di voler scatenare la terza guerra mondiale?". L'icona offre informazioni visive sull'importanza della finestra di dialogo. Il numero e il tipo di pulsanti di comando può variare da uno a tre.

La finestra di dialogo più semplice è quella che visualizza un messaggio sullo schermo e fornisce un pulsante di comando OK, in modo che l'utente possa far sparire la finestra.

Il codice BASIC di seguito riportato crea la semplice finestra di dialogo illustrata nella Figura 13.2:

```
Private Sub Form_Load()
  MsgBox "Acquistate un Macintosh", ,"Windows si è bloccato
un'altra volta"
End Sub
```

Questa semplice finestra di dialogo non fa altro che comparire sullo schermo e quindi scomparire quando l'utente fa clic su OK.

Figura 13.2 Un messaggio di errore

Aggiungere una icona alla casella di messaggio

Le icone possono essere utili per attirare l'attenzione di un utente sulla finestra di dialogo. Visual Basic offre quattro icone (Figura 13.3).

Icona	Nome	Valore numerico	Costante Visual Basic
	Critical Message	16	vbCritical
	Warning Query	32	vbQuestion
	Warning Message	48	vbExclamation
	Information Message	64	vbInformation

Figura 13.3 Le quattro icone per le finestre di dialogo e i corrispondenti valori numerici

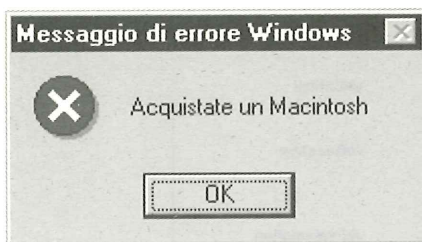
- **L'icona Critical Message:** l'icona *Critical Message* segnala all'utente un problema di estrema importanza, quale: "Continuando, cancellerai tutti i file presenti su disco fisso. Sei sicuro di volerlo fare?".
- **L'icona Warning Query:** l'icona *Warning Query* (un punto di domanda) evidenzia problemi meno gravi, come: "Vuoi davvero uscire da Microsoft Word?".
- **L'icona Warning Message:** l'icona *Warning Message* (un punto esclamativo) sottolinea degli avvertimenti che l'utente deve assolutamente notare, come: "Stai per sostituire tutte le 79 pagine del documento con un punto!".
- **L'icona Information Message:** l'icona *Information Message* conferisce un aspetto interessante a messaggi altrimenti noiosi, come: "La stampa di tutte le 3.049 pagine del documento potrebbe essere lunga. Fai clic su OK se vuoi proseguire".

Per aggiungere un'icona a una finestra di dialogo, va inserito il valore numerico corrispondente all'icona che si desidera visualizzare tra il messaggio della finestra di dialogo e il testo della barra del titolo nel seguente modo:

```
Private Sub Form_Load()
  MsgBox "Acquistate un Macintosh", 16, "Messaggio di errore
Windows"
End Sub
```

Queste tre linee di codice hanno come effetto quello di visualizzare la casella di messaggio illustrata nella Figura 13.4.

Solitamente Visual Basic permette di visualizzare soltanto una delle quattro possibili icone in una casella di messaggio. Se ciò vi sembra limitativo, potete sempre creare una finestra di dialogo personalizzata partendo da zero. A questo scopo create una form separata, impostate la proprietà BorderStyle a Fixed Dialog e disegnate i pulsanti di comando e un riquadro di testo direttamente su questa form; dopodiché potrete disegnare tutti i riquadri di immagine che volete e caricare in ognuno l'icona che più preferite.

Figura 13.4 Una finestra di dialogo che contiene l'icona Critical Message

Si ricordi solo che creare una finestra di dialogo utilizzando una form separata richiede la definizione di pulsanti di comando, etichette e riquadri di immagine e soprattutto la stesura del codice BASIC che faccia funzionare il tutto. Se si vuole creare una casella di messaggio in modo rapido e semplice, è possibile utilizzare il comando `MsgBox`.

Definire il numero e il tipo di pulsanti di comando in una casella di messaggio

Le caselle di messaggio possono contenere da uno a tre pulsanti di comando. Ogni pulsante di comando è rappresentato da un valore numerico; nella Tabella 13.1 sono elencate le sei combinazioni di pulsanti di comando.

Per comunicare a Visual Basic di utilizzare una determinata combinazione di pulsanti di comando, si scelga la combinazione desiderata e si digiti il corrispondente valore numerico tra il testo del messaggio e il testo della barra del titolo nel seguente modo:

```
Private Sub Form_Load()
  MsgBox "File non trovato", 2, "Messaggio di errore"
End Sub
```

oppure

```
Private Sub Form_Load()
  MsgBox "File non trovato", vbAbortRetryIgnore, "Messaggio di
errore"
End Sub
```

Quale pulsante di comando ha selezionato l'utente nella casella di messaggio?

Se una finestra di dialogo si limita a visualizzare il pulsante di comando OK, in genere facendo clic su OK si ottiene la scomparsa della finestra. Al contrario, le

Tabella 13.1 Le combinazioni di comandi disponibili in Visual Basic

Visualizza	*Valore*	*Costante di Visual Basic*
Pulsante OK	0	vbOKOnly
Pulsanti OK e Annulla	1	vbOKCancel
Pulsanti Termina, Riprova e Ignora	2	vbAbortRetryIgnore
Pulsanti Sì, No e Annulla	3	vbYesNoCancel
Pulsanti Sì e No	4	vbYesNo
Pulsanti Riprova e Annulla	5	vbRetryCancel

finestre di dialogo con due o più pulsanti di comando offrono all'utente la possibilità di scegliere. Quando vengono inseriti più pulsanti di comando su una finestra di dialogo, occorre scrivere il codice BASIC necessario per capire:

- ✓ quale pulsante è stato selezionato dall'utente
- ✓ che cosa deve fare il programma in risposta al pulsante di comando selezionato.

I sette pulsanti di comando che un utente può selezionare sono rappresentati dai valori numerici della Tabella 13.2.

Tabella 13.2 Pulsanti di comando che possono essere selezionati dall'utente

Pulsante selezionato	*Valore numerico*	*Costante di Visual Basic*
OK	1	vbOK
Annulla	2	vcCancel
Termina	3	vbAbort
Riprova	4	vbRetry
Ignora	5	vbIgnore
Sì	6	vbYes
No	7	vbNo

Per fare in modo che il programma determini quale pulsante di comando è stato selezionato dall'utente, si dovrà assegnare a una variabile il valore di ritorno della funzione `MsgBox`, come illustrato di seguito:

```
Risposta = MsgBox("File non trovato", 2, "Messaggio di errore")
```

Questa linea di codice visualizza una finestra di dialogo con i pulsanti di comando Termina, Riprova e Ignora. Se l'utente fa clic su Termina, il valore di Risposta è 3. Se l'utente fa clic su Riprova, il valore di Risposta è 4. Se l'utente fa clic su Ignora, il valore di Risposta è 5.

Nota. Ogni volta che si utilizza una variabile per memorizzare il valore corrispondente al pulsante selezionato si dovranno racchiudere tra parentesi tonde gli argomenti della funzione `MsgBox`.

Le finestre di dialogo preimpostate

Sebbene in molti casi le caselle di messaggio siano più che sufficienti, ci sono situazioni che richiedono finestre di dialogo più complesse. Per questo motivo, oltre alla possibilità di visualizzare caselle di messaggio, Visual Basic offre la possibilità di utilizzare delle finestre di dialogo preimpostate per le funzioni più comuni.

- ✓ Apri
- ✓ Salva con nome
- ✓ Colore
- ✓ Carattere
- ✓ Stampa.

Prima di poter utilizzare queste finestre di dialogo, è necessario caricare l'icona CommonDialog nella casella degli strumenti di Visual Basic. A questo scopo, si utilizzi la procedura di seguito riportata.

1. **Selezionate Progetto-Componenti oppure premete CTRL+T.**

   La finestra di dialogo Componenti viene visualizzate.

2. **Fate clic sulla scheda Controlli.**
3. **Controllate che sia presente un segno di spunta nella casella di controllo Microsoft CommonDialog Control 6.0; se il segno di spunta non fosse presente, fate clic nella casella di controllo.**
4. **Fate clic su OK.**

   La casella degli strumenti di Visual Basic visualizza l'icona CommonDialog, come illustrato nella Figura 13.5.

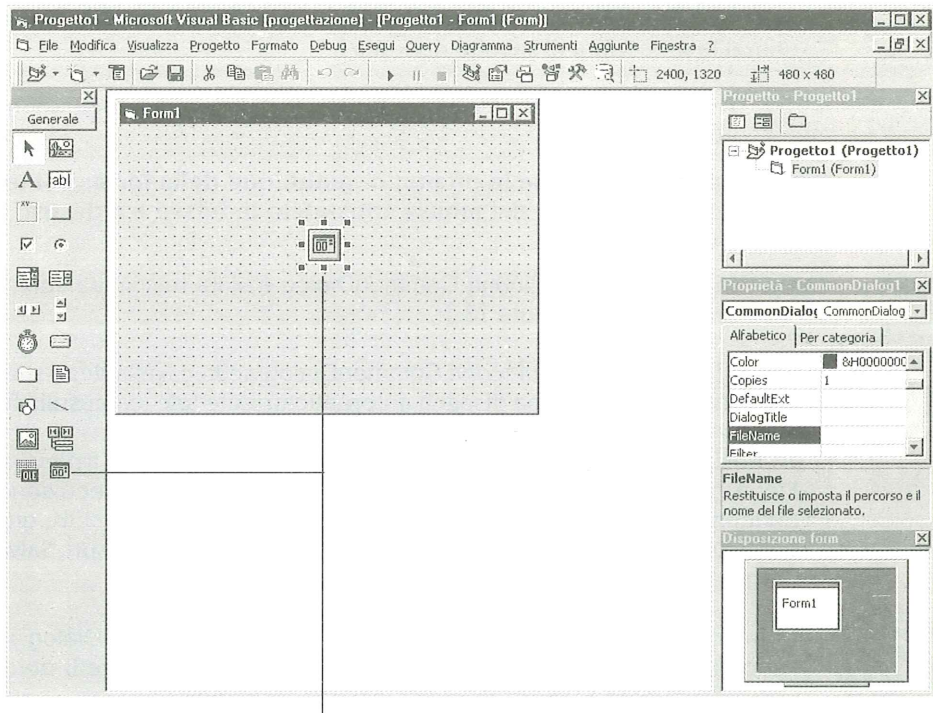



Figura 13.5 L'icona CommonDialog nella casella degli strumenti e su una form

Mettete alla prova le nuove conoscenze

1. **Perché visualizzare un'icona in una finestra di dialogo?**

   a. Nel caso un utente analfabeta desideri utilizzare il vostro programma.

   b. Per attirare l'attenzione dell'utente e fornire un indizio visivo. Per esempio, un'icona Critical Message può avvertire l'utente che potrebbe verificarsi qualcosa di terribile se non verranno adottate misure adeguate.

   c. Per vedere se l'utente è abbastanza intelligente da accorgersi che la finestra di dialogo non ha nulla di importante da comunicare.

2. **Per utilizzare una finestra di dialogo Apri, Salva con nome, Colore, Carattere o Stampa, che cosa dovete fare come prima cosa?**

   a. Acquistare i manuali che in passato venivano forniti gratuitamente insieme al programma.

   b. Creare una nuova form, disegnare tre pulsanti di comando, due caselle di controllo, una casella di riepilogo e una pernice su un pero.

   c. Salvare il file e uscire da Visual Basic.

   d. Controllare di avere trascinato l'icona Common Dialog sulla form.

Per visualizzare una di queste cinque finestre di dialogo nel vostro programma, si utilizzi la procedura di seguito riportata.

1. **Fate clic sull'icona CommonDialog nella casella degli strumenti di Visual Basic.**
2. **Spostate il mouse in un punto qualunque della form; poi tenendo premuto il pulsante del mouse spostatelo in basso e a destra. Rilasciate il pulsante.**

   In questo modo l'icona CommonDialog è stata trascinata sulla form, come illustrato nella Figura 13.5.

Facendo doppio clic sull'icona CommonDialog nella casella degli strumenti di Visual Basic, il programma disegna automaticamente tale icona sulla form.

La posizione dell'icona CommonDialog sulla form è del tutto irrilevante, dato che l'icona rimane sempre invisibile quando il programma è in esecuzione. Inserendo un'icona CommonDialog su una form, si dice a Visual Basic: "OK, questa icona ti conferisce poteri magici per visualizzare le finestre di dialogo Apri, Salva con nome, Stampa, Colore o Carattere ogni volta che te lo dirò".

Dato che per ogni form è necessaria una sola icona CommonDialog, non è necessario cambiare il nome dell'icona, ma è sufficiente utilizzare il nome di default che Visual Basic assegna all'icona, ovvero CommonDialog1.

Visualizzare la finestra di dialogo Apri

La finestra di dialogo Apri (Figura 13.6) permette agli utenti di selezionare un'unità, una directory e un file da aprire. L'utente ha inoltre la possibilità di visualizzare solo tipi particolari di file, come quelli che corrispondono ai criteri *.TXT oppure *.EXE.

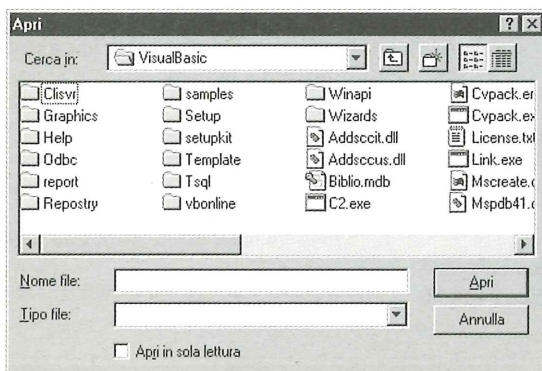



Figura 13.6 La finestra di dialogo Apri

Per visualizzare la finestra di dialogo Apri si ha bisogno di un solo comando BASIC, analogo al seguente:

```
CommonDialog1.ShowOpen
```

Per definire l'elenco dei file visualizzati nella finestra di dialogo Apri, si deve utilizzare un elemento tecnico definito filtro, che comunica a Visual Basic i tipi di file da visualizzare, per esempio tutti quelli con l'estensione TXT o BAT.

Un filtro comprende due elementi: l'etichetta che appare nella casella di riepilogo e il filtro stesso. Nella Tabella 13.3 sono riportati alcuni esempi di etichette e di filtri. Per maggiore chiarezza, le etichette comprendono solitamente, oltre alla descrizione, anche il filtro al quale sono associate.

I file di testo hanno per esempio l'estensione TXT, ma talvolta hanno anche l'estensione ASC. Di conseguenza, l'etichetta "File di testo (TXT)" comunica che la finestra di dialogo visualizza soltanto i file di testo con l'estensione TXT e non i file di testo con l'estensione ASC.

Per definire le etichette e i filtri, va utilizzato il codice BASIC come illustrato nel seguente esempio:

```
CommonDialog1.Filter = "Tutti i file (*.*) ¦ *.* ¦ File di testo
            (*.TXT) ¦ *.TXT ¦ File batch
            (*.BAT) ¦ *.BAT ¦ File eseguibili
(*.EXE) ¦ *.EXE
```

Questo comando definisce quattro filtri e associa ad ognuno un numero di identificazione progressivo. Nell'esempio precedente, il filtro Tutti i file è il primo e quindi ha il numero 1.

Dopo avere definito il filtro, va stabilito quale filtro deve essere utilizzato per default. Anche in questo caso, si utilizzi il codice BASIC:

```
CommonDialog1.FilterIndex = 1
```

Tabella 13.3 Etichette e filtri

Etichetta	*Filtro*
Tutti i file (*.*)	*.*
File di testo (*.TXT)	*.TXT
File batch (*.BAT)	*.BAT
File eseguibili (*.EXE)	*.EXE

Questa riga di codice visualizza "Tutti i file (*.*)" nella casella di riepilogo Tipi di file. Nel caso si desideri utilizzare un filtro diverso, per esempio il quarto, è possibile digitare quanto segue:

```
CommonDialog1.FilterIndex = 4
```

Questa riga visualizza "File eseguibili (*.EXE)" nella casella di riepilogo Tipi di file.

L'ordine in base al quale definite il filtro (utilizzando il comando CommonDialog1. Filter), determina il numero FilterIndex. Per esempio, se si modificasse il filtro nel seguente modo:

```
CommonDialog1.Filter = "File di testo (*.TXT) ¦ *.TXT ¦ Tutti
i file (*.*) ¦ *.*"
```

il seguente codice visualizzerebbe "File di testo (*.TXT)" nella casella di riepilogo Tipi di file:

```
CommonDialog1.FilterIndex = 1
```

Creare una finestra di dialogo Apri

Ovviamente, se il programma dovrà memorizzare dei dati in un file, prima o poi dovrà anche poter aprire un file. Dato che quasi tutti i programmi devono aprire i file, è possibile semplificare la questione utilizzando la finestra di dialogo Apri.

Per creare una finestra di dialogo Apri, dovete scrivere un codice BASIC analogo al seguente:

```
Private Sub mnuFileApri_Click()
  CommonDialog1.Filter = "File di testo (*.TXT) ¦ *.TXT ¦
           Tutti i file (*.*) ¦ *.*"
  CommonDialog1.FilterIndex = 1
  CommonDialog1.ShowOpen
End Sub
```

Questa procedura di gestione degli eventi comunica a Visual Basic quanto segue.

1. Quando l'utente fa clic sul comando Apri nel menu a discesa File, occorre seguire le istruzioni racchiuse tra la prima e l'ultima riga della procedura `Private Sub mnuFileApri_Click()`.
2. La seconda riga indica a Visual Basic quali filtri rendere disponibili nella casella combinata Tipo file della finestra Apri.
3. La terza riga indica a Visual Basic di utilizzare come default il primo filtro e quindi visualizzare solo i file con estensione `.TXT`.

4. La quarta riga indica a Visual Basic di visualizzare la finestra di dialogo Apri sullo schermo.

A questo punto, la finestra di dialogo Apri ha un aspetto gradevole e sembra funzionare ma, dato che non è stato scritto il codice BASIC che ne elabora i risultati, il suo utilizzo è ancora inutile.

Quale file ha selezionato l'utente nella finestra di dialogo Apri?

Dopo che è stata visualizzata la finestra di dialogo Apri, è necessario scoprire quale file l'utente ha selezionato. Quando l'utente fa clic su un file visualizzato nella finestra di dialogo Apri, Visual Basic memorizza il nome del file nella proprietà Filename dell'oggetto CommonDialog. Quindi, nel caso si voglia recuperare il nome file sul quale l'utente ha fatto clic, va utilizzata una variabile per memorizzare la proprietà Filename nel seguente modo:

```
NomeDelFile = CommonDialog1.filename
```

La procedura completa potrebbe essere simile a questa:

```
Private Sub mnuFileApri_Click()
Dim NomeDelFile As String
  CommonDialog1.Filter = "File di testo (*.TXT) ¦ *.TXT ¦
            Tutti i file (*.*) ¦ *.*"
  CommonDialog1.FilterIndex = 1
  CommonDialog1.ShowOpen
  NomeDelFile= CommonDialog1.filename
End Sub
```

La proprietà `CommonDialog1.filename` comprende sia il nome del file, sia la directory in cui il file è memorizzato, come C:\MieiDocumenti\Segreti\Resume.txt. Se l'utente fa clic sul pulsante di comando Annulla nella finestra di dialogo Apri, la proprietà filename verrà impostata su "" (assenza di testo).

Visualizzare la finestra di dialogo Salva con nome

La finestra di dialogo Salva con nome è quasi identica alla finestra di dialogo Apri. Tuttavia il testo della barra del titolo non è uguale, dato che la barra del titolo della finestra di dialogo Apri dice "Apri", mentre la barra del titolo della finestra di dialogo Salva con nome dice "Salva con nome".

L'unico comando BASIC che va utilizzato per visualizzare la finestra di dialogo Salva con nome è:

```
CommonDialog1.ShowSave
```

Tuttavia è anche possibile utilizzare i filtri per visualizzare determinati tipi di file, per esempio:

```
Private Sub mnuFileSalvaCome_Click()
  CommonDialog1.Filter = "File di testo (*.TXT) ¦ *.TXT ¦
            Tutti i file (*.*) ¦ *.*"
  CommonDialog1.FilterIndex = 1
  CommonDialog1.ShowSave
End Sub
```

Quale file ha selezionato l'utente nella finestra di dialogo Salva con nome?

Anche la finestra di dialogo Salva con nome, proprio come quella Apri, memorizza il nome del file nella proprietà Filename dell'oggetto CommonDialog. Quindi, se si vuole recuperare il nome file sul quale l'utente ha fatto clic, va utilizzata una variabile per memorizzare la proprietà Filename nel seguente modo:

```
NomeDelFile = CommonDialog1.filename
```

La procedura completa potrebbe essere simile a questa:

```
Private Sub mnuFileSalvaCome_Click()
Dim NomeDelFile As String
  CommonDialog1.Filter = "File di testo (*.TXT) ¦ *.TXT ¦
            Tutti i file (*.*) ¦ *.*"
  CommonDialog1.FilterIndex = 1
  CommonDialog1.ShowSaveAs
  NomeDelFile= CommonDialog1.filename
End Sub
```

La finestra di dialogo Salva con nome non salva i file da sola. Per salvare un file con un nome differente dovrete scrivere altro codice BASIC che dica al computer di salvare un file su un disco.

Invece di utilizzare le finestre di dialogo Apri e Salva con nome di Visual Basic, provate con un controllo File che offre maggiore flessibilità e un più elevato numero di funzioni, pur essendo più piccolo da utilizzare. Potete scaricare questo controllo FileDialog dal sito Web CCRP all'indirizzo `www.mvps.org/ccrp` (questo controllo FileDialog è stato creato da un gruppo indipendente di programmatori, quindi se non funziona non arrabbiatevi con Microsoft).

Visualizzare la finestra di dialogo Colore

La finestra di dialogo Colore permette agli utenti di selezionare dei colori oppure di crearne di personalizzati, come illustrato nella Figura 13.7. Per visualizzare la finestra di dialogo Colore si dovranno utilizzare due comandi BASIC:

```
CommonDialog1.Flags = cdlCCRGBInit
CommonDialog1.ShowColor
```

La prima riga indica a Visual Basic che l'oggetto CommonDialog, il cui nome è CommonDialog1, può essere utilizzato per recuperare un colore dalla finestra di dialogo Colore.

La seconda riga indica a Visual Basic di visualizzare la finestra di dialogo Colore.

Per esempio:

```
Private Sub mnuStrumentiColore_Click()
  CommonDialog1.Flags = cdlCCRGBInit
  CommonDialog1.ShowColor
End Sub
```

Quale colore ha selezionato l'utente nella finestra di dialogo Colore?

La finestra di dialogo Colore memorizza il colore selezionato dall'utente nella proprietà Color della finestra di dialogo Common.

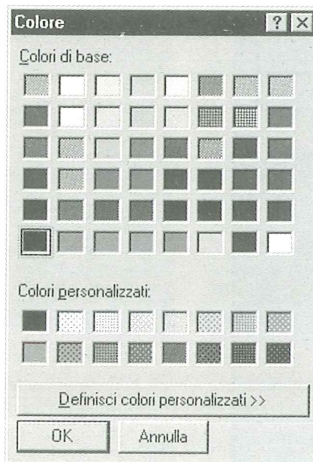



Figura 13.7 La finestra di dialogo Colore

Quindi, nel caso si desideri recuperare il colore (che Visual Basic memorizza come numero) sul quale l'utente ha fatto clic, va impostata una variabile per memorizzare la proprietà Color nel seguente modo:

```
Colore = CommonDialog1.Color
```

Quindi la procedura completa potrebbe essere simile a questa:

```
Private Sub cmdColore_Click()
Dim Colore As Long
  CommonDialog1.Flags = cdlCCRGBInit
  CommonDialog1.ShowColor
  Colore = CommonDialog1.Color
End Sub
```

Visualizzare la finestra di dialogo Carattere

La finestra di dialogo Carattere, illustrata nella Figura 13.8, permette agli utenti di selezionare diversi caratteri, stili e dimensioni in punti. Ogni volta che l'utente seleziona un'opzione, questa finestra di dialogo visualizza un esempio, in modo che gli utenti possano verificare se il carattere, lo stile o le dimensioni in punti corrispondono alle loro esigenze.

Per visualizzare la finestra di dialogo Carattere, si devono utilizzare due comandi BASIC:

```
CommonDialog1.Flags = cdlCFEffects Or cdlFBoth
CommonDialog1.ShowFont
```

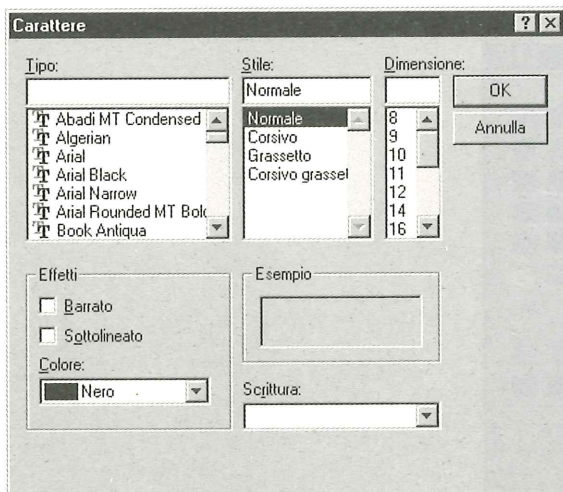



Figura 13.8 La finestra di dialogo Carattere

La prima riga indica a Visual Basic, il cui nome è `CommonDialog1`, che l'oggetto CommonDialog può essere utilizzato per recuperare informazioni sul tipo di carattere che l'utente ha scelto nella finestra di dialogo Carattere.

La seconda riga indica a Visual Basic di visualizzare la finestra di dialogo Carattere. Per esempio:

```
Private Sub mnuFormatoCarattere_Click()
  CommonDialog1.Flags = cdlCFEffects Or cdlFBoth
  CommonDialog1.ShowFont
End Sub
```

Quali opzioni ha selezionato l'utente nella finestra di dialogo Carattere?

La finestra di dialogo Carattere permette all'utente di selezionare diverse opzioni, come illustrato nella Tabella 13.4.

Quindi la procedura completa di gestione degli eventi potrebbe essere simile alla seguente:

```
Private Sub mnuFormatoCarattere_Click()
Dim TestoColore As Long
Dim Grassetto As Boolean
Dim Corsivo As Boolean
Dim Sottolineato As Boolean
Dim Barrato As Boolean
Dim Carattere As String
Dim Dimensione As Integer
  CommonDialog1.Flags = cdlEffects Or cdlFBoth
  CommonDialog1.ShowFont
  TestoColore = CommonDialog1.Color
  Grassetto = CommonDialog1.FontBold
  Corsivo = CommonDialog1.FontItalic
  Sottolineato = CommonDialog1.FontUnderline
  Barrato = CommonDialog1.Strikethru
  Carattere = CommonDialog1.FontName
  Dimensione = CommonDialog1.FontSize
End Sub
```

Visualizzare la finestra di dialogo Stampa

La finestra di dialogo Stampa (Figura 13.9) permette agli utenti di selezionare la stampante, l'intervallo di stampa e il numero di copie.

Tabella 13.4 Proprietà che memorizzano i valori dalla finestra di dialogo Carattere

Proprietà	*Informazioni contenente*
Color	Il colore selezionato; per utilizzare questa proprietà, dovete innanzitutto impostare a cdlEffects la proprietà Flags.
FontBold	True se è stato impostato il grassetto, False in caso contrario.
FontItalic	True se è stato impostato il corsivo, False in caso contrario.
FontStrikethru	True se è stato impostato il barrato, False in caso contrario. Per utilizzare questa proprietà, dovete innanzitutto impostare a cdlEffects la proprietà Flags.
FontUnderline	True se è stato impostato il sottolineato, False in caso contrario. Per utilizzare questa proprietà, dovete innanzitutto impostare a cdlEffects la proprietà Flags.
FontName	Il nome del tipo di carattere selezionato.
FontSize	Le dimensioni selezionate del tipo di carattere.

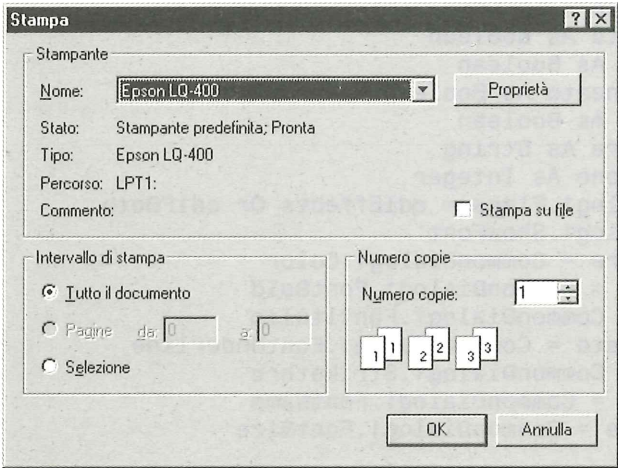



Figura 13.9 La finestra di dialogo Stampa

Per visualizzare la finestra di dialogo Stampa, si utilizzi il seguente comando BASIC:

```
CommonDialog1.ShowPrinter
```

Per esempio:

```
Private Sub mnuFileStampa_Click()
  CommonDialog1.ShowPrinter
End Sub
```

Per impostare il valore di default per il numero di copie da stampare, dovete utilizzare i seguenti comandi BASIC:

```
CommonDialog1.Copies = 1
CommonDialog1.ShowPrinter
```

Per esempio:

```
Private Sub mnuFileStampa_Click()
  CommonDialog1.Copies = 1
  CommonDialog1.ShowPrinter
End Sub
```

In modo analogo alle finestre di dialogo Apri, Salva con nome, Colore e Carattere, la finestra di dialogo Stampa sembra funzionare, ma in realtà non farà nulla se prima non verrà scritto il codice BASIC che elabori le informazioni raccolte.

Servendovi delle finestre di dialogo illustrate nel presente capitolo potrete conferire al vostro programma quell'aspetto e quell'aria professionale che la gente ormai si aspetta dal software. Come qualsiasi programmatore professionista potrà confermarvi, più il vostro programma sembra essere in grado di funzionare, maggiori sono le probabilità che la gente creda che il programma funzioni davvero.

Parte IV

I principi di base per la scrittura del codice

"E ALLORA HO DETTO, 'CAMERIERE! CAMERIERE! C'È UN BACO NELLA MINESTRA!"
E LUI MI HA RISPOSTO, 'MI DISPIACE, SIGNORE, È IL CAPO A PROGRAMMARE I COMPUTER'
AHH HAHA HAHA. GRAZIE! GRAZIE!"

In questa parte...

Urrà! Ecco il primo capitolo dove finalmente si comincia a scrivere il codice BASIC necessario per indurre il vostro computer a fare qualcosa di utile. Fino a questo momento, avete solo disegnato gli elementi che compongono un'interfaccia utente (inserendovi di tanto in tanto un comando BASIC). Tuttavia si sa che l'aspetto non basta. L'importante è che la vostra interfaccia utente sia non solo bella, ma anche in grado di rispondere all'utente.

Sebbene l'idea di scrivere del codice BASIC possa intimidire, in realtà non è nulla di complicato. Il codice BASIC non è altro che una serie di istruzioni utilizzate per comunicare passo dopo passo al computer che cosa fare. Quindi preparatevi a *codificare* (un termine dei programmatori per indicare la scrittura dei comandi per il computer). Scoprirete che la programmazione può essere veramente divertente, facile e affascinante quanto disegnare un'interfaccia utente o fare un gioco sul computer.

Capitolo 14

Procedure di gestione degli eventi

In questo capitolo

- Creare le procedure di gestione degli eventi
- Visualizzare diverse procedure di gestione degli eventi
- Apportare delle modifiche nella finestra Codice

Ogni volta che l'utente compie un'azione, come fare clic con il mouse, premere un tasto, svenire sulla tastiera oppure sparare un colpo al monitor, si verifica un *evento*. Nell'attimo in cui accade l'evento, Visual Basic cerca il codice BASIC che dica al programma che cosa fare. Il codice BASIC che risponde a un evento specifico viene definito *procedura di gestione di quell'evento*.

Un singolo programma Visual Basic può contenere diverse migliaia di procedure di gestione degli eventi.

Se ne possedete così tante significa che avete un programma eccezionalmente complesso oppure che siete dei programmatori del tutto incompetenti. Con tanti eventi possibili e altrettante possibili procedure di gestione degli eventi in un solo programma, come fa Visual Basic a sapere quale procedura utilizzare?

La risposta è semplice. Quando si verifica un evento, di solito esso è indirizzato a un componente dell'interfaccia utente. Per esempio, la maggior parte degli utenti fa clic con il pulsante del mouse solo quando il mouse sta puntando un oggetto, come un pulsante di comando, una casella di controllo oppure un comando di menu visualizzato sullo schermo.

Qualunque oggetto può disporre di una o più procedure di gestione degli eventi e ciascuna procedura risponde a diversi eventi, come un clic del mouse o la pressione di un tasto.

Tipi di eventi

I tipi di eventi più comuni possono essere classificati in tre categorie.

- ✓ **Gli eventi della tastiera** si verificano quando l'utente preme un determinato tasto, come TAB, o una particolare combinazione di tasti, come CTRL+P.
- ✓ **Gli eventi del mouse** si verificano quando l'utente sposta il mouse, fa clic o doppio clic con il pulsante del mouse oppure trascina il mouse sullo schermo.
- ✓ **Gli eventi del programma** si verificano quando un programma Visual Basic fa qualcosa da solo come caricare, aprire o chiudere una form. Mentre gli eventi della tastiera e del mouse si verificano quando l'utente fa qualcosa, gli eventi del programma si verificano quando il codice BASIC deve fare qualcosa.

Nonostante Visual Basic riesca a rispondere a una serie di eventi, di solito è sufficiente che l'interfaccia utente risponda a uno o due eventi, come un clic del mouse o la pressione di un determinato tasto. Non appena individua un evento, Visual Basic cerca di capire quale elemento dell'interfaccia utente deve rispondere.

Per esempio, quando l'utente fa clic con il mouse, Visual Basic identifica innanzitutto l'evento ("OK, era un clic del mouse"), poi cerca di capire dove si è verificato questo clic ("L'utente ha fatto clic sul pulsante di comando OK").

Infine Visual Basic individua la procedura di gestione degli eventi collegata a quel particolare pulsante di comando; la procedura in questione contiene il codice BASIC che comunica al programma che cosa fare quando l'utente fa clic con il pulsante del mouse.

Creare le procedure di gestione degli eventi

Un oggetto può rispondere a uno o più eventi. Per esempio, un pulsante di comando può rispondere quando l'utente fa clic con il pulsante del mouse oppure preme INVIO.

Due o più oggetti possono rispondere allo stesso evento. Per esempio, sia un pulsante di comando che una casella di controllo possono rispondere a un clic del mouse; tuttavia è possibile che due oggetti abbiano istruzioni diverse che dicono a Visual Basic che cosa fare in seguito.

Per scrivere una procedura di gestione degli eventi, è necessario eseguire i passaggi riportati alla pagina successiva.

- ✓ Identificare l'elemento dell'interfaccia utente che deve reagire.
- ✓ Aprire la finestra Codice.
- ✓ Identificare l'evento a cui rispondere.
- ✓ Scrivere il codice BASIC per elaborare l'evento.

Prima di creare qualsiasi procedura di gestione degli eventi, è necessario controllare che tutti gli oggetti dell'interfaccia utente abbiano un nome. Se si crea una procedura di questo tipo per un oggetto e in seguito si modifica il nome dell'oggetto, dovrà essere riscritta l'intera procedura.

Questi tre elementi dell'interfaccia utente che possono essere associati a eventi:

- ✓ le form
- ✓ gli oggetti (pulsanti di comando, caselle di controllo e così via)
- ✓ i menu a discesa.

Per creare una procedura di gestione degli eventi per una form, si utilizzi la procedura di seguito riportata.

1. **Fate clic su un punto qualsiasi della form, ma non sugli oggetti presenti nella form.**
2. **Aprite la finestra Codice. Aprite la finestra Codice premendo F7, selezionando Visualizza-Codice oppure facendo doppio clic su un punto qualsiasi della form (ma non sugli oggetti presenti nella form).**

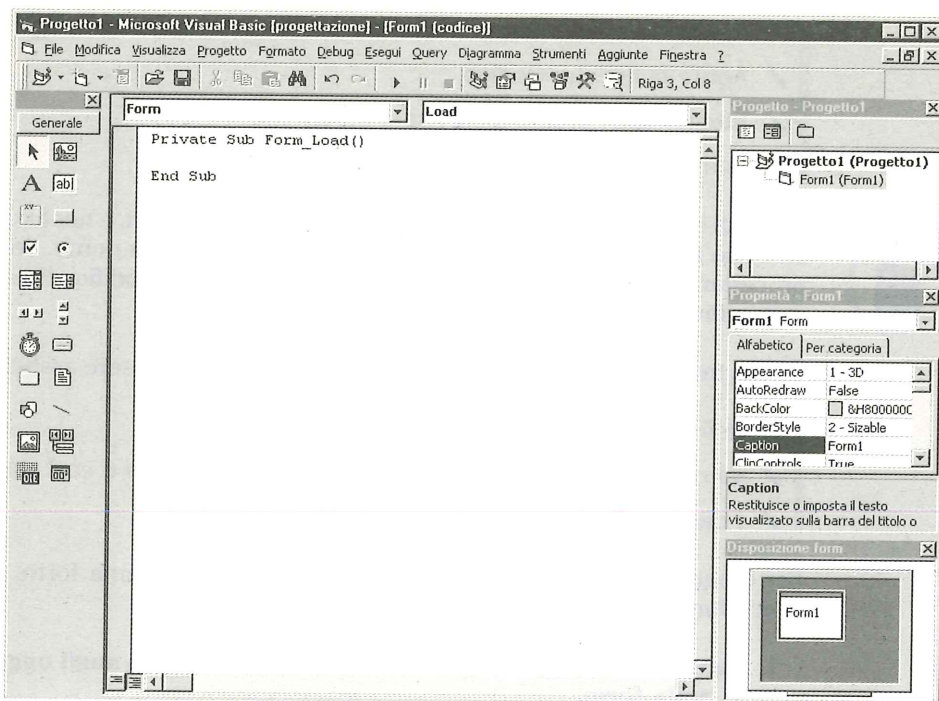
   Visual Basic visualizza la finestra Codice insieme a una procedura di gestione degli eventi vuota (Figura 14.1).

Per creare una procedura di gestione degli eventi per un oggetto, si utilizzi la procedura di seguito riportata.

1. **Fate clic su un oggetto in modo che attorno a esso compaiano dei rettangolini neri (maniglie).**
2. **Aprite la finestra Codice. Aprite la finestra Codice premendo F7, selezionando Visualizza-Codice oppure facendo doppio clic sull'oggetto.**

   Visual Basic visualizza la finestra Codice insieme a una procedura di gestione degli eventi vuota. Potrebbe essere necessario fare clic anche sulla casella di riepilogo Procedura per selezionare un evento specifico a cui rispondere, come Clic o KeyPress.

Con un doppio clic su un oggetto (come un pulsante di comando), Visual Basic visualizza immediatamente la finestra Codice.

Figura 14.1 La finestra Codice con la struttura di una procedura di gestione degli eventi vuota

Per creare una procedura di gestione degli eventi per un menu a discesa, procedete nella maniera seguente.

1. Fate clic sul titolo del menu a discesa contenente il comando di menu desiderato.
2. Fate clic sul comando di menu per il quale intendete scrivere il codice BASIC.

Visual Basic visualizzerà la finestra Codice insieme a una procedura di gestione degli eventi vuota.

Le parti delle procedure di gestione degli eventi

Quando create una procedura di gestione degli eventi per la prima volta, Visual Basic visualizza una procedura di gestione degli eventi vuota nella finestra Codice. Tutte le procedure di questo tipo sono costituite da due righe, come:

```
Private Sub cmdUscita_Click()
End Sub
```

La prima riga di qualsiasi procedura di gestione degli eventi comprende cinque parti.

- `Private Sub`: identifica la procedura come un sottoroutine;
- **il nome dell'oggetto:** in questo esempio l'oggetto è denominato `cmdUscita`;
- **un trattino di sottolineatura**;
- **il nome dell'evento:** in questo esempio l'evento è un clic del mouse.
- **una coppia di parentesi che contiene tutti i dati di cui la sottoroutine potrebbe aver bisogno per funzionare:** in questo esempio le parentesi sono vuote, per indicare che non sono necessari altri dati.

La procedura di gestione degli eventi precedente dice al computer: "Ecco le istruzioni da seguire ogni volta che l'utente fa clic con il mouse sul pulsante di comando cmdUscita. Adesso lasciatemi in pace".

Dato che questo esempio non contiene istruzioni da seguire, questa procedura di gestione degli eventi non fa assolutamente nulla, proprio come alcuni dei vostri colleghi.

Ogni volta che si modifica il nome di un oggetto, dovrà essere modificato anche il nome di tutte le procedure di gestione degli eventi collegate all'oggetto stesso. In caso contrario, Visual Basic non saprà quali procedure appartengano agli oggetti presenti sulla vostra interfaccia utente.

Suddividere in due la finestra Codice

Dopo che si sono cominciate a scrivere numerose procedure di gestione degli eventi, la finestra Codice potrebbe non essere in grado di visualizzare tutte le procedure contemporaneamente. Se si desidera visualizzare due o più procedure di gestione degli eventi, è possibile suddividere a metà la finestra Codice in modo orizzontale. La finestra Codice può essere suddivisa solo a metà (non in quarti, in terzi e così via).

Per suddividere a metà la finestra Codice si utilizzi la procedura di seguito riportata.

1. **Posizionate il puntatore del mouse sulla barra di divisione.**

   Quest'ultima appare in cima alla barra di scorrimento verticale. Non appena il puntatore del mouse si trova sopra la barra di divisione, si trasforma in due righe parallele orizzontali con due frecce rivolte verso l'alto e verso il basso.

2. **Tenete premuto il pulsante sinistro del mouse e trascinate il mouse verso il basso.**

Quando la barra di divisione avrà diviso in due la finestra Codice nel modo desiderato, rilasciate il pulsante del mouse (Figura 14.2).

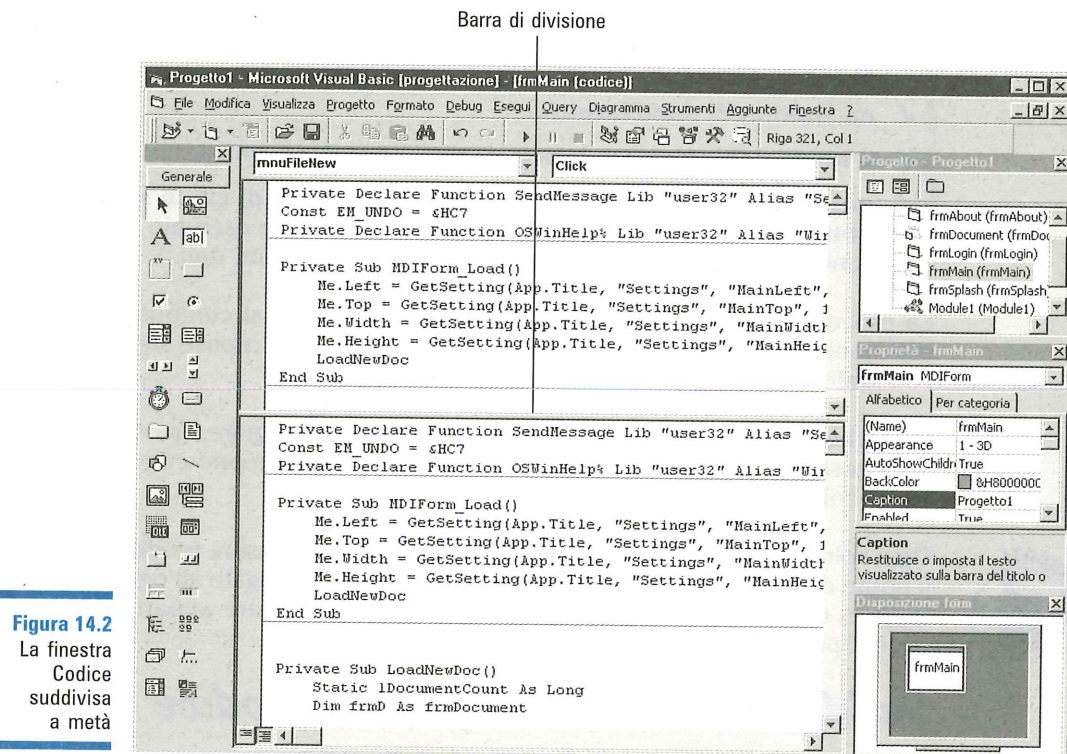



Figura 14.2 La finestra Codice suddivisa a metà

Per tornare a visualizzare la finestra Codice come finestra singola, si utilizzi la procedura di seguito riportata.

1. **Posizionate il puntatore del mouse sulla barra che divide a metà la finestra Codice.**

   Il puntatore del mouse si trasforma in due righe parallele orizzontali con due frecce rivolte verso l'alto e verso il basso.

2. **Tenete premuto il pulsante sinistro del mouse e trascinate il mouse verso l'alto fino in cima (o in fondo) alla finestra Codice; dopodiché rilasciate il pulsante del mouse.**

Apportare delle modifiche nella finestra Codice

La finestra Codice funziona come un semplice programma di elaborazione testi. Nella Tabella 14.1 sono elencati i diversi comandi che possono essere utilizzati per modificare le procedure di gestione degli eventi.

Tabella 14.1 Tasti di modifica comuni

Tasto	*Che cosa succede*
CANC	Cancella il carattere a destra del cursore.
BACKSPACE	Cancella il carattere a sinistra del cursore.
CTRL+Y	Cancella la riga sulla quale si trova il cursore.
⇱	Sposta il cursore all'inizio della riga.
FINE	Sposta il cursore alla fine della riga.
CTRL+⇱	Sposta il cursore sulla prima procedura di gestione degli eventi.
CTRL+FINE	Sposta il cursore sull'ultima procedura di gestione degli eventi.
CTRL+↓	Visualizza la procedura successiva.
CTRL+↑	Visualizza la procedura precedente.
CTRL+PAG.↓	Va alla prima riga della procedura seguente
CTRL+PAG.↑	Va alla prima riga della procedura corrente o di quella precedente.
MAIUSC+F2	Se il cursore appare sul nome di una procedura, questo comando visualizza il codice BASIC relativo a quella procedura.
CTRL+→	Sposta il cursore di una parola a destra.
CTRL+←	Sposta il cursore di una parola a sinistra.
PAG ↓	Visualizza la pagina successiva nella finestra Codice.
PAG↑	Visualizza la pagina precedente nella finestra Codice.
INS	Attiva o disattiva la modalità inserimento.
CTRL+X	Taglia il blocco di testo selezionato.
CTRL+C	Copia il blocco di testo selezionato.

continua

Tabella 14.1

Tasto	*Che cosa succede*
CTRL+V	Incolla il blocco di testo tagliato o copiato in precedenza.
CTRL+Z	Annulla l'ultima azione compiuta (una lettera digitata, una frase cancellata e così via).
CTRL+F	Trova una parola specificata.
F1	Visualizza la guida di Visual Basic.
F3	Trova la successiva occorrenza della parola specificata in precedenza con il comando CTRL+F.
MAIUSC+F3	Trova l'ultima occorrenza della parola specificata in precedenza con il comando CTRL+F.
F6	Passa tra i riquadri della finestra Codice (se la finestra è suddivisa a metà).
CTRL+H	Ricerca una parola e la sostituisce con qualcos'altro.
CTRL+P	Visualizza la finestra di dialogo Stampa.

Per aiutarvi a scrivere il codice BASIC, la finestra Codice evidenzia automaticamente le parole chiave riservate BASIC. In questo modo, è facile distinguere le parole chiave riservate BASIC dai comandi che avete creato voi.

Per cancellare un'intera procedura degli eventi si evidenzi la procedura utilizzando il mouse o i tasti di direzione e si prema CANC.

Visualizzare diverse procedure di gestione degli eventi

Un tipico programma Visual Basic consiste di procedure di gestione degli eventi memorizzate nei file FRM. Per trovare una particolare procedura da esaminare o modificare, si hanno due possibilità:

- ✓ selezionare un nome di oggetto dalla casella di riepilogo Oggetto e poi selezionare un evento dalla casella di riepilogo Procedura nella finestra Codice.
- ✓ utilizzare la finestra Visualizzatore oggetti.

Selezionare una procedura di eventi con le caselle di riepilogo Oggetto e Procedura

Per utilizzare le caselle di riepilogo Oggetto e Procedura al fine di trovare una procedura di gestione degli eventi, è necessario conoscere il nome del file FRM nel quale è stata memorizzata la procedura. La casella di riepilogo Oggetto contiene tutti gli oggetti memorizzati su una form. Facendo clic su questa casella, è possibile trovare l'oggetto che contiene la procedura di gestione degli eventi che si desidera visualizzare.

La casella di riepilogo Procedura contiene tutti gli eventi ai quali un oggetto può rispondere. Ogni volta che viene selezionato un evento diverso, Visual Basic visualizza una procedura degli eventi diversa nella finestra Codice.

Per visualizzare una procedura di gestione degli eventi con le caselle di riepilogo Oggetto e Procedura, si eseguano questi passaggi.

1. **Utilizzate uno dei seguenti metodi per attivare la finestra Progetto:**
   - fate clic nella finestra Progetto con il mouse;
   - premete CTRL+R.
   - selezionate Visualizza-Gestione progetti.
2. **Fate clic sulla form che contiene la procedura di gestione degli eventi che desiderate.**
3. **Fate clic sull'icona Visualizza codice nella finestra Progetto oppure selezionate Visualizza-Codice.**
4. **Fate clic sulla freccia rivolta verso il basso nella casella di riepilogo Oggetto e poi fate clic sul nome dell'oggetto.**
5. **Fate clic sulla freccia rivolta verso il basso nella casella di riepilogo Procedura e poi fate clic sul nome dell'evento.**

   Visual Basic visualizza la procedura di gestione degli eventi prescelta.

Se una procedura di gestione degli eventi contiene il codice BASIC, la casella di riepilogo Procedura visualizza il nome dell'evento in grassetto, come illustrato nella Figura 14.3. Se invece un nome dell'evento appare con lo stile normale, significa che quella procedura è vuota.

Selezionare una procedura di eventi con la finestra Visualizzatore oggetti

La finestra Visualizzatore oggetti è utile quando volete visualizzare le procedure di gestione degli eventi memorizzate in file diversi. Per visualizzare una proce-

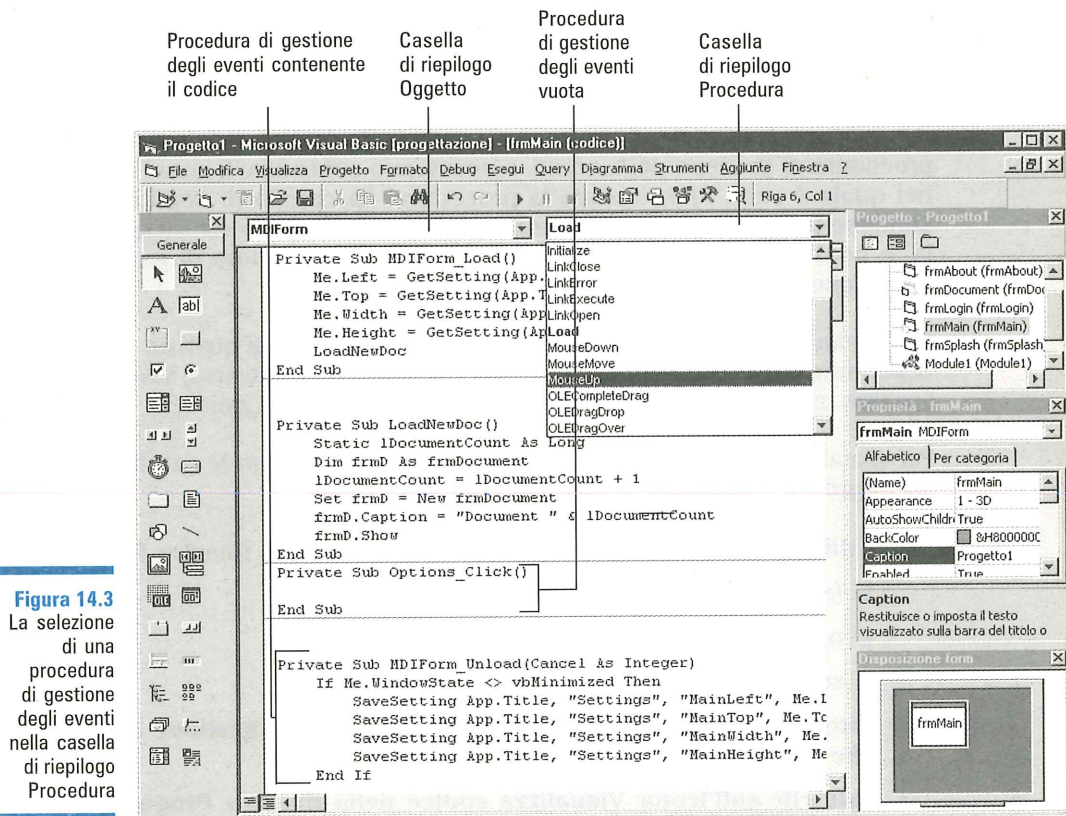



Figura 14.3 La selezione di una procedura di gestione degli eventi nella casella di riepilogo Procedura

dura di gestione degli eventi con la finestra Visualizzatore oggetti, utilizzate la procedura di seguito riportata.

1. **Selezionate Visualizza-Visualizzatore oggetti oppure premete F2 per visualizzare la finestra Visualizzatore oggetti, (Figura 14.4).**

2. **Fate clic sulla freccia rivolta verso il basso nella casella di riepilogo Progetto/Libreria e poi fate clic sul progetto che contiene la procedura di gestione degli eventi che volete visualizzare.**

3. **Fate clic sul nome file della form nel riquadro di sinistra Classi.**

   Tutte le procedure di gestione degli eventi memorizzate in quel file vengono visualizzate in grassetto sul riquadro di destra.

4. **Fate doppio clic sulla procedura di gestione degli eventi che appare nel riquadro di destra.**

Figura 14.4 La finestra di dialogo Visualizzatore oggetti

La Tabella 14.2 elenca gli eventi più comuni ai quali gli oggetti possono rispondere.

Tabella 14.2 Eventi comuni

Evento	*Si verifica quando...*
Activate	Una form si trasforma in una finestra attiva.
Change	Cambia il contenuto di una casella combinata, di una casella di riepilogo directory, di una casella di riepilogo unità, di una barra di scorrimento, di un'etichetta, di un riquadro di disegno o di una casella di testo.
Click	L'utente fa clic con il pulsante del mouse su un oggetto.
DblClick	L'utente fa doppio clic con il pulsante del mouse su un oggetto.

continua

Tabella 14.2 Eventi comuni

Evento	*Si verifica quando...*
Deactivate	Una form passa dallo stato di finestra attiva a quello di finestra inattiva.
DragDrop	L'utente tiene premuto il pulsante del mouse su un oggetto, sposta il mouse e poi rilascia il pulsante.
DragOver	L'utente tiene premuto il pulsante del mouse su un oggetto e poi sposta il mouse.
DropDown	La parte di elenco di una casella combinata si espande verso il basso per visualizzare un elenco di opzioni.
GotFocus	Un oggetto viene evidenziato quando l'utente preme TAB oppure fa clic su un oggetto.
KeyDown	L'utente preme un tasto.
KeyPress	L'utente preme e rilascia un tasto ANSI, come un carattere della tastiera, una combinazione di tasti con CTRL, INVIO oppure BACKSPACE (in linea di principio, un tasto ANSI può essere rappresentato da qualsiasi lettera, numero o combinazione di tasti).
KeyUp	L'utente rilascia un tasto.
LostFocus	Un oggetto smette di essere evidenziato perché l'utente ha premuto TAB o ha fatto clic su un altro oggetto oppure se una form viene scaricata.
MouseDown	L'utente preme un pulsante del mouse.
MouseMove	L'utente sposta il mouse.
MouseUp	L'utente rilascia un pulsante del mouse.

La combinazione del nome dell'oggetto e del nome dell'evento definisce il nome della procedura di gestione degli eventi. Dato che i nomi degli oggetti devono essere unici, due procedure di gestione degli eventi sulla stessa form non possono avere lo stesso nome.

Sebbene due procedure di gestione degli eventi sulla stessa form non possano condividere lo stesso nome, procedure di gestione degli eventi su form diverse possono avere lo stesso nome. Per esempio, potreste avere un pulsante di comando chiamato `cmdExit` che compare su due diverse form.

In questo caso, la seguente procedura di gestione degli eventi potrebbe essere memorizzata su entrambe le form:

```
Private Sub cmdUscita_Click()
End Sub
```

Due o più oggetti possono condividere lo stesso nome se li trasformate in un *array di controllo*. Per il momento non è necessario che memorizziate questo termine; è sufficiente prenderne nota e continuare. Se davvero desiderate saperne di più sugli array di controllo, procuratevi una copia dell'ultima edizione di *More Visual Basic For Dummies*.

La procedura di gestione degli eventi di cui ogni programma ha bisogno

La procedura di gestione degli eventi più semplice e più importante necessaria a tutti i programmi è quella che serve a interrompere il programma.

La procedura di seguito riportata comunica a Visual Basic di interrompere l'esecuzione del programma quando l'utente fa clic su un pulsante di comando denominato `cmdUscita`.

```
Private Sub cmdUscita_Click()
  Unload Me
End Sub
```

Se non si inserisce una procedura di gestione degli eventi per interrompere un programma, l'unico sistema a cui l'utente può ricorrere è riavviare il computer oppure spegnerlo. Dato che questo non è il metodo migliore, non si dimentichi mai di inserire nel programma almeno un sistema (se non di più) che permetta all'utente di uscire in qualunque momento.

Il sistema precedente per interrompere un programma Visual Basic consisteva nell'utilizzare la parola chiave `End` nel seguente modo:

```
Private Sub cmdUscita_Click()
End
End Sub
```

Microsoft consiglia comunque di utilizzare il comando `Unload Me` al posto della parola chiave `End`. Questo comando però interrompe un programma Visual Basic soltanto se tutte le form del programma sono state scaricate utilizzando il comando `Unload`. Se anche una sola delle form non è stata scaricata, il comando `Unload Me` non interrompe l'esecuzione di un programma Visual Basic.

Mettete alla prova le vostre nuove conoscenze

1. Che cos'è un evento e quali sono i tre tipi di evento?

a. Un evento è qualcosa per cui bisogna acquistare dei biglietti, come un concerto, una manifestazione sportiva e il circo.

b. Gli eventi sono cose che accadono al computer, per esempio una bibita versata sulla tastiera, la cancellazione di tutti i file per errore oppure il cane che mangia un dischetto.

c. Gli eventi si verificano quando l'utente preme un tasto o il pulsante del mouse oppure quando il programma modifica il proprio aspetto. I tre tipi di evento sono gli eventi della tastiera, del mouse e del programma.

d. Un evento è una vacanza o una cerimonia che consente di lasciare il lavoro per una giornata. I tre tipi di evento sono le ferie, le riunioni e i funerali dei parenti non stretti.

2. Che cosa fanno la casella di riepilogo Oggetto e la casella di riepilogo Procedura nella finestra Codice?

a. Elencano le possibili ragioni che possono spiegare perché fareste meglio a scrivere il programma in C++ o in Pascal piuttosto che in Visual Basic.

b. La casella di riepilogo Oggetto permette di selezionare un oggetto per il quale scrivere una procedura. La casella di riepilogo Procedura permette di scegliere tutti i possibili eventi cui un oggetto può rispondere.

c. La casella di riepilogo Oggetto contiene un elenco di tutti gli oggetti non acuminati che potete utilizzare per colpire il computer. La casella di riepilogo Procedura elenca tutti gli eventi ai quali fareste meglio a presenziare invece di restare seduti davanti al computer.

d. Nessuna delle due fa qualcosa che valga la pena di ricordare, quindi non preoccupatevi di rispondere nuovamente a questa domanda.

Capitolo 15

Utilizzare le variabili

In questo capitolo

- Utilizzare le variabili
- Assegnare numeri e stringhe alle variabili
- Dichiarare i tipi di dati

Quando si sarà deciso che cosa far fare al programma, sarà giunto il momento di scrivere il codice BASIC. Il primo codice che va scritto si trova all'interno delle procedure di gestione degli eventi.

Al livello più semplice una procedura di gestione degli eventi dice al computer che cosa fare. Per esempio, una procedura di questo tipo per uscire dal programma fornisce al computer un'unica istruzione `Unload Me`:

```
Private Sub cmdUscita_Click()
  Unload Me
End Sub
```

Questa procedura di gestione degli eventi non richiede alcuna informazione all'utente, a parte il semplice evento rappresentato da un clic del pulsante di comando `cmdUscita`.

Che cosa succede, tuttavia, quando un utente digita un nome, un indirizzo o un numero telefonico nel programma? Ovviamente, il programma deve leggere queste informazioni dall'interfaccia utente e poi deve utilizzarle in qualche modo.

Il CD-ROM contiene un semplice programma che vi mostra come cancellare, usare, e mostrare una stringa variabile in una casella di testo o differenti forms.

Leggere i dati

Tutte le informazioni che vengono inviate a un programma dall'esterno vengono definite *dati*. Quasi tutti i programmi, tranne forse i più semplici, ricevono dei dati, li utilizzano e quindi li rimandano fuori.

Un programma di elaborazione testi riceve i propri dati sotto forma di parole; queste parole vengono formattate dal programma per dare loro un aspetto gradevole, quindi il programma le stampa ordinatamente su carta. Un database riceve i propri dati sotto forma di nomi, indirizzi e numeri telefonici. Il database memorizza le informazioni in qualche luogo e poi visualizza i dati in un formato considerato utile dall'utente. Un sistema di guida per missili nucleari riceve i propri dati sotto forma di coordinate del bersaglio.

Qualunque programma utile segue queste tre fasi fondamentali.

1. Recupero dei dati.
2. Elaborazione dei dati.
3. Restituzione dei dati.

Qualunque programma inutile presenta queste quattro caratteristiche:

- è troppo difficile da utilizzare
- costa un sacco di soldi
- sostiene di essere "user-friendly"
- non funziona.

Lo scopo di un programma è quello di trasformare i computer in tritatutto elettrici. Si infilano le informazioni da una parte e si aspetta che escano dall'altra. A prescindere dal programma in questione (programma di elaborazione testi, foglio elettronico, database o videogioco), tutti i programmi gestiscono quanto segue:

- numeri
- stringhe.

I *numeri* possono essere positivi, negativi, numeri interi, frazioni oppure qualsiasi altro numero che siate in grado di immaginare (compresi i numeri telefonici per appuntamenti da cardiopalmo, i numeri che compongono la combinazione della cassaforte contenente una valanga di soldi e i numeri immaginari comprensibili solo a cervelloni della matematica).

Le *stringhe* sono caratteri legati insieme. Un *carattere* è rappresentato da qualsiasi cosa venga digitata mediante la tastiera, tra cui lettere, simboli di punteggiatura e (non fate confusione adesso) numeri.

A seconda di come il programma decide di trattarli, i numeri possono essere considerati come numeri oppure come stringhe. Per esempio, la maggior parte dei programmi considera il vostro numero di telefono o l'indirizzo come una stringa, ma tratta la vostra età e il vostro peso come un numero.

Una singola lettera viene considerata una stringa. Anche una frase completa è una stringa. Persino il primo capitolo di *Guerra e Pace* può essere considerato come una stringa. Le stringhe possono essere un insieme qualsiasi di lettere, spazi e numeri.

Valori e variabili

Quando si digita un numero o una stringa in un programma, in che modo il computer tiene traccia di quel numero o di quella stringa? Dopotutto, voi potreste anche sapere che 02-4563234 rappresenta un numero telefonico ma per il computer si tratta semplicemente di un altro numero o di un'altra stringa.

Per memorizzare i dati, i programmi utilizzano le *variabili*, vale a dire un concetto algebrico consolidatosi nel tempo. Quando scrivete un programma dovete dirgli: "Allora, quando qualcuno digita 02-4563234, attribuisci a questo numero un nome, come NumeroTel e registrarlo in un posto dove sia possibile ritrovarlo".

Quando il programma dovrà recuperare o utilizzare questi dati, dirà: "OK, dove ho messo queste informazioni? Ah, sì, sono in un posto (una variabile) denominato NumeroTel". Il computer si precipiterà obbediente alla variabile NumeroTel e recupererà qualunque numero o parola si trovi lì.

Le variabili possono contenere una vasta gamma di dati (ecco perché vengono definite variabili). Le informazioni memorizzate in una variabile sono definite *valori*, indipendentemente dal fatto che si tratti di una stringa o di un numero.

Utilizzare le variabili

Esistono due tipi di variabile:

- ✓ quelle inventate da voi
- ✓ quelle già definite come le proprietà di ciascun oggetto presente sulla form.

Ogni volta che si disegna un oggetto per creare l'interfaccia utente, Visual Basic crea automaticamente una serie completa di variabili (definite *proprietà*) impostate a valori di default. Per vedere i valori delle proprietà di un oggetto, va utilizzata la finestra Proprietà (premendo F4 oppure selezionando Visualizza-Finestra Proprietà).

I valori delle proprietà possono rappresentare numeri (come la definizione della larghezza e dell'altezza di un oggetto), valori True o False (per esempio per definire se un oggetto deve essere visibile o meno) o stringhe (come i titoli di un pulsante di comando). Le proprietà si limitano a definire l'aspetto di un oggetto sullo schermo.

Le variabili sono nomi che possono rappresentare qualsiasi tipo di valore, mentre le proprietà sono nomi speciali per le variabili che influiscono sull'aspetto di un oggetto.

Per creare una variabile personale, è sufficiente che le venga attribuito un nome: dopo che si sarà digitato il nome di una variabile, questa comincerà a esistere come per magia. Per creare una variabile esistono due metodi:

- ✓ dichiararla utilizzando l'istruzione `Dim`
- ✓ limitarsi ad assegnarle un nome e un valore.

L'unico luogo in cui è possibile digitare (e creare) il nome di una variabile è la finestra Codice e, più precisamente, tra la prima e l'ultima riga di una procedura.

Dichiarare le variabili

Per creare una variabile è sufficiente digitarne il nome e attribuirle un valore, come illustrato nel seguente esempio:

```
Private Sub Form1_Load()
  NomeMicia = "Olimpia la gatta"
  Anni = 6
End Sub
```

La procedura di gestione degli eventi precedente dice quanto segue.

1. Quando viene caricata una form denominata `Form1`, queste sono le istruzioni da seguire.
2. Innanzitutto, creare una variabile denominata `NomeMicia` e impostarne il valore alla stringa "Olimpia la gatta".
3. Secondo, creare una variabile denominata `Anni` e impostarne il valore al numero 6.

Sebbene sia del tutto accettabile creare una variabile dal nulla nel bel mezzo di una procedura, questo comportamento non è considerato una valida tecnica di programmazione. Senza esaminare una procedura di gestione degli eventi riga dopo riga non è possibile rendersi conto di quante variabili siano contenute in essa.

Una tecnica di programmazione migliore consiste nel dichiarare le variabili all'inizio di qualsiasi procedura di gestione degli eventi. Per dichiarare una variabile, si utilizzi il comando `Dim`, come illustrato nel seguente esempio:

```
Dim NomeVariabile1, NomeVariabile2
```

Si digitino tutti i nomi delle variabili desiderate nel comando precedente.

Riscrivendo la procedura di gestione degli eventi precedente, si ottiene quanto segue:

```
Private Sub Form1_Load()
Dim NomeMicia, Anni
  NomeMicia = "Olimpia la gatta"
  Anni = 6
End Sub
```

È anche possibile dichiarare la variabile in maniera esplicita, in questo modo:

```
Dim NomeMicia as String, Anni as Integer
```

Questa tecnica viene illustrata dettagliatamente nei paragrafi "Tipi di dati".

Sebbene la dichiarazione di una variabile possa aumentare di una riga o due la dimensione di una procedura di gestione degli eventi, avete visto quanto è facile trovare i nomi delle variabili utilizzate? Invece di esaminare una procedura di gestione degli eventi riga per riga, è sufficiente dare un'occhiata alle prime righe per visualizzare un elenco dei nomi di variabile utilizzati.

L'utilizzo dell'istruzione `Dim` può aiutare i programmatori a capire la funzione svolta da una procedura. Per quanto riguarda i computer, il fatto che l'istruzione `Dim` venga utilizzata o meno non ha alcuna importanza. Alcuni esempi di variabili sono riportati nella Figura 15.1.

Attribuire un nome alle variabili

È possibile attribuire alle variabili un nome qualsiasi e memorizzare al loro interno tutto quello che si vuole.

Comunque, nominando una variabile `PhoneNumber` e poi sostituire l'indirizzo di qualcuno in una variabile è piuttosto semplice.

Per utilizzarle al meglio, attribuite alle vostre variabili dei nomi che rappresentino i dati che volete memorizzare. Per esempio, il fatto di chiamare `NumeroTel` una variabile ha senso solo se si ha intenzione di memorizzarvi dei numeri di telefono. In modo analogo, una variabile chiamata `NomiLavoro` non dovrebbe contenere altro che nomi di colleghi di lavoro.

Quando attribuite un nome alle variabili dovete rispettare alcune regole ferree; in caso contrario Visual Basic si arrabbierà. Tutte le variabili devono:

- ✓ cominciare con una lettera;

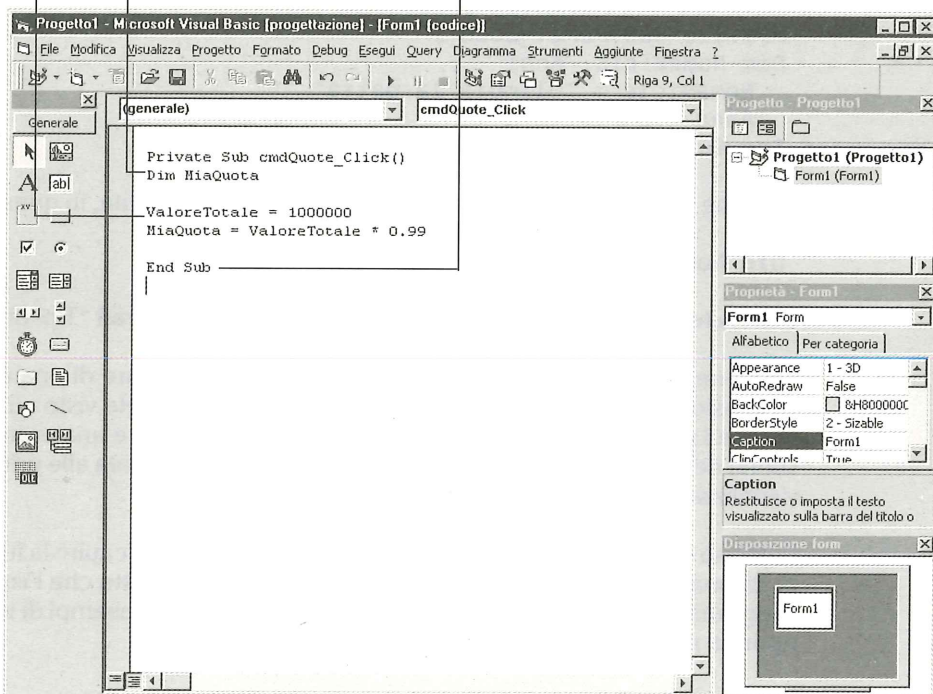



Figura 15.1 Due metodi per creare una variabile

- avere una lunghezza massima di 255 caratteri (con il minimo evidente di un carattere);
- contenere solo lettere, numeri e il trattino di sottolineatura (_); gli spazi e i simboli di punteggiatura non sono ammessi.
- essere diverse dalle parole riservate di Visual Basic, come `End` e `Sub`.

Se i nomi soddisfano questi criteri, tutto filerà liscio; ovviamente questo non significa che il programma funzionerà ma per lo meno Visual Basic sarà felice. Di seguito sono riportati degli esempi di nomi di variabili approvate da Visual Basic:

```
Telefono
Questo_è_il_tuo_nome
Statale66
```

Mettete alla prova le vostre nuove conoscenze

1. Che cosa fanno i seguenti comandi BASIC?

```
txtMessaggio.TXT="Sto
cancellando il disco fisso"

cmdModifica.Caption = "Ha, ha,
ha!"

Gotcha = 1
```

a. Immette la stringa "Sto cancellando il disco fisso" nella proprietà Text di una casella di testo denominata txtMessaggio, memorizza la stringa "Ha, ha, ha!" nella proprietà Caption di un pulsante di comando denominato cmdModifica e imposta a 1 una variabile denominata `Gotcha`.

b. Cancella il disco fisso una volta, visualizza il messaggio "Ha, ha, ha!" e provoca un grave danno psicologico all'utente.

c. Memorizza la stringa `"1"` in una variabile denominata `Gotcha` e quindi cancella il disco fisso come punizione per aver scelto questa risposta che è chiaramente sbagliata.

d. Questi comandi BASIC non fanno assolutamente nulla se non accendete il computer e caricate Visual Basic. Ecco tutto.

2. Qual è lo scopo della dichiarazione delle variabili?

a. In questo modo i programmatori possono sentirsi a loro agio parlando al computer; sono sicuri che li capisce perfettamente.

b. Perché così potete nascondere il fatto che in realtà non sapete che cosa state facendo.

c. Per fornire un comodo elenco di tutte le variabili e definirle utilizzando particolari tipi di dati. Come interi o stringhe.

d. Per liberarle dalle tenebre e portarle alla luce, dove potranno essere etichettate, rintracciate e distrutte.

Questi invece sono nomi di variabili non accettabili, che verranno quindi rifiutati da Visual Basic:

`123Sorpresa`	(questo nome comincia con un numero)
`Solo lavoro`	(il nome comprende uno spazio)
`Sub`	(questo nome è una parola chiave riservata di Visual Basic).

Attribuire dei numeri alle variabili

Ora che si è in grado di creare delle variabili attribuendo loro dei nomi, come fare ad assegnare un valore a una variabile e recuperare questo valore? Facile: è sufficiente utilizzare un elemento misterioso denominato segno di uguale (=).

Per attribuire un valore a una variabile, si deve scrivere un comando BASIC come nel seguente esempio:

```
NomeVariabile = Valore
```

Invece di dire al computer: "Ehi, stupido. Assegna il numero 36 a una variabile denominata `Anni`", ci si può limitare a scrivere:

```
Anni = 36
```

Le variabili possono contenere solo un valore per volta. Se una variabile possiede già un valore e gliene viene attribuito un altro, questa si limita a buttare fuori il valore vecchio e accettare quello nuovo. Si supponga di dare due comandi, come nel seguente esempio:

```
Anni = 36
Anni = 49
```

Visual Basic all'inizio direbbe: "OK, la mia variabile denominata `Anni` contiene il numero `36`". Poi Visual Basic darebbe un'occhiata alla seconda riga e direbbe: "OK, ora la mia variabile denominata `Anni` contiene il numero `49` e mi dimenticherò totalmente che il numero 36 sia mai esistito".

Dato che le proprietà di un oggetto sono variabili, è possibile assegnare dei valori a un oggetto utilizzando lo stesso metodo. Per esempio, supponete di voler modificare la proprietà `Height` di una casella di testo denominata `txtPassword` impostandola a `1200`. Ecco come procedere:

```
txtPassword.Height = 1200
```

In questo modo si dice a Visual Basic: "Trova l'oggetto denominato `txtPassword` e modifica la sua proprietà `Height` impostandola a `1200`".

Se si desidera essere più precisi, è possibile persino fare così:

```
frmSegreto!txtPassword.Height = 1200
```

Questa istruzione dice a Visual Basic: "Sulla form denominata `frmSegreto`, trova l'oggetto che si chiama `txtPassword` e modifica la sua proprietà `Height` impostandola a `1200`".

Se non viene inserito il nome della form, Visual Basic suppone che l'oggetto desiderato si trovi sulla form che contiene il codice BASIC che si sta scrivendo. Per recuperare il valore dalla proprietà di un oggetto (per esempio per scoprire l'altezza di una casella di testo), attribuite una variabile a detta proprietà, così:

```
Dim ButtonHeight As Integer
  ButtonHeight = txtMangiaQuesto.Height
```

1. Create una variabile chiamata ButtonHeight utilizzando il tipo Integer.
2. Alla variabile chiamata ButtonHeight attribuite il valore memorizzato nella proprietà Height della casella di testo chiamata txtMangiaQuesto.

Attribuire delle stringhe alle variabili

L'attribuzione di stringhe alle variabili è simile all'attribuzione di numeri; l'unica differenza sta nel fatto che si deve chiudere la stringa tra virgolette, in modo che Visual Basic riesca a identificarne l'inizio e la fine.

Per esempio, si attribuirà alla variabile la seguente stringa composta da un'unica parola:

```
Nome = "Gianni"
```

Oppure si attribuirà alla variabile una stringa comprendente due o più parole:

```
Nome = "Gianni Bianchi"
```

oppure

```
Nome = "Gianni Bianchi, il migliore"
```

Non tutte le stringhe sono costituite da lettere. Talvolta è possibile attribuire a una variabile di tipo stringa un numero telefonico o il numero del libretto della mutua nel seguente modo:

```
NumeroTel = "02-4563234"
LibrettoMutua = "123-45-6789"
```

Che cosa succede se si dimenticano le virgolette e si scrive semplicemente così:

```
NumeroTel = 02-4563234
LibrettoMutua = 123-45-6789
```

Senza virgolette, Visual Basic considera il trattino come una sottrazione e pensa che si intenda calcolare un risultato. Invece di memorizzare 02-4563234 nella variabile `NumeroTel`, Visual Basic visualizzerebbe -4563232, mentre nella variabile `LibrettoMutua` visualizzerebbe -6711.

Il principio fondamentale per l'assegnazione delle variabili è il seguente: quando si attribuiscono a una variabile lettere o numeri che si vogliono considerare come stringa, queste lettere e questi numeri vanno racchiusi tra virgolette.

Modificare le proprietà

Il fatto di assegnare a una variabile dei numeri o delle stringhe non costituisce un arido esercizio senza alcuna relazione effettiva con la vita reale. Per esempio, volendo visualizzare un messaggio sullo schermo, vanno modificate le proprietà di un'etichetta o di una casella di testo. Se si desidera creare un'animazione, vanno modificate costantemente le proprietà Left e Top che definiscono la posizione di un oggetto sullo schermo. Dato che le proprietà di un oggetto sono variabili, si può modificare un oggetto attribuendo nuovi valori alle sue proprietà.

Supponiamo che due pulsanti di comando e le caselle di testo, illustrate nelle Figure 15.2 e 15.3, abbiano le proprietà di seguito riportate.

Oggetto	*Pmroprietà*	*Valore*
Casella di testo	Name	txtMessaggio
	Text	(vuoto)
Pulsante di comando in alto	Name	cmdCiao
	Caption	Ciao
Pulsante di comando in basso	Name	cmdAddio
	Caption	Addio

Ora supponiamo che il primo pulsante di comando abbia la seguente procedura di gestione degli eventi:

```
Private Sub cmdCiao_Click()
  txtMessaggio.Text = "Ciao, mondo!"
End Sub
```

E il secondo pulsante di comando abbia la seguente procedura di gestione degli eventi:

```
Private Sub cmdAddio_Click()
  txtMessaggio.Text = "Addio, mondo crudele!"
End Sub
```

Quando si fa clic sul pulsante di comando Ciao, ecco che cosa succede.

1. Visual Basic individua l'evento Click e nota che il mouse sta puntando a un pulsante di comando denominato `cmdCiao`.

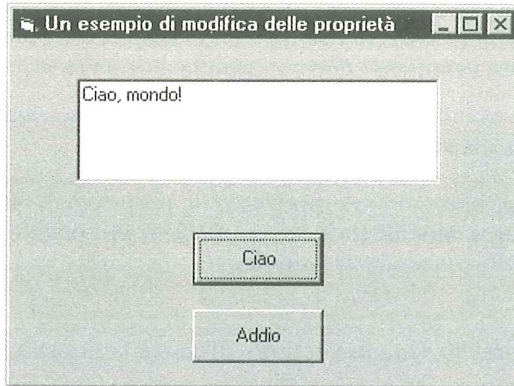



Figura 15.2 Un clic sul pulsante di comando Ciao

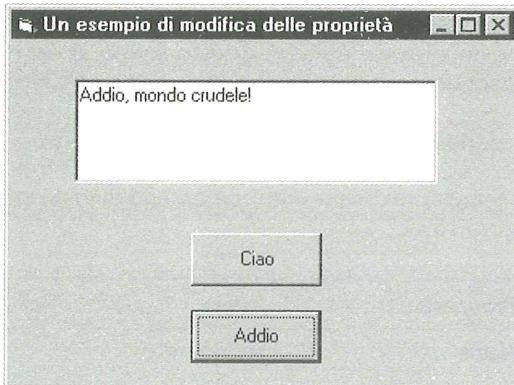



Figura 15.3 Un clic sul pulsante di comando Addio

2. Visual Basic individua rapidamente la procedura denominata `cmdCiao_Click()` e cerca ulteriori istruzioni.

3. La procedura di gestione degli eventi `cmdCiao_Click()` dice a Visual Basic: "Trova una casella di testo denominata `txtMessaggio` e sostituisci la sua proprietà Text con la stringa `"Ciao, mondo!"`.

4. La stringa `"Ciao, mondo!"` appare all'interno della casella di testo denominata `txtMessaggio` (Figura 15.2).

Quando si fa clic sul pulsante di comando Addio, ecco che cosa succede.

1. Visual Basic individua l'evento Click e nota che il mouse sta puntando a un pulsante di comando denominato `cmdAddio`.

2. Visual Basic individua rapidamente la procedura denominata `cmdAddio_Click()` e cerca ulteriori istruzioni.

3. La procedura di gestione degli eventi `cmdAddio_Click()` dice a Visual Basic: "Trova una casella di testo denominata `txtMessaggio` e sostituisci la sua proprietà Text con la stringa `"Addio, mondo crudele!"`.
4. La stringa `"Addio, mondo crudele!"` appare all'interno della casella di testo denominata `txtMessaggio` (Figura 15.3).

Il codice BASIC può essere utilizzato per modificare le proprietà di tutti gli oggetti che compaiono su una form. Modificando le proprietà di altri oggetti, è possibile visualizzare messaggi e informazioni per l'utente.

L'unica proprietà che il codice BASIC non è in grado di modificare è la proprietà Name degli oggetti; per fare ciò, sarà necessario utilizzare la finestra Proprietà.

Attribuire delle variabili ad altre variabili

Oltre ad assegnare numeri e stringhe a una variabile, è anche possibile assegnare il valore di una variabile a un'altra variabile. A questo scopo si deve scrivere un comando BASIC simile al seguente:

```
NomePrimaVariabile = NomeSecondaVariabile
```

Per esempio, considerate l'aggiunta di una seconda casella di testo con le proprietà di seguito riportate (Figura 15.4).

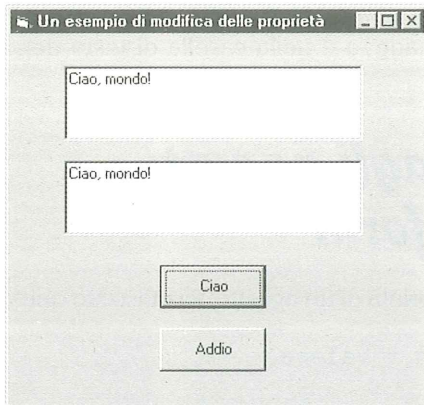
Oggetto	*Proprietà*	*Valore*
Seconda casella di testo	Name	txtCopione
	Text	(vuoto)

Prendete in considerazione le seguenti modifiche alla procedura di gestione degli eventi `cmdCiao_Click()`:

```
Private Sub cmdCiao_Click()
  txtMessaggio.Text = "Ciao, mondo!"
  txtCopione.Text = txtMessaggio.Text
End Sub
```

e alla procedura di gestione degli eventi `cmdAddio_Click()`:

```
Private Sub cmdAddio_Click()
  txtMessaggio.Text = "Addio, mondo crudele!"
  txtCopione.Text = txtMessaggio.Text
End Sub
```

Figura 15.4 L'aggiunta di una seconda casella di testo

Ora, quando si fa clic sul pulsante di comando Ciao, ecco che cosa succede.

1. Visual Basic individua l'evento Click e nota che il mouse sta puntando a un pulsante di comando denominato `cmdCiao`.
2. Visual Basic individua rapidamente la procedura denominata `cmdCiao_Click()` e cerca ulteriori istruzioni.
3. La procedura di gestione degli eventi `cmdCiao_Click()` dice a Visual Basic: "Trova una casella di testo denominata `txtMessaggio` e sostituisci la sua proprietà `Text` con la stringa `"Ciao, mondo!"`".
4. Visual Basic nota la seconda istruzione che dice: "Trova una casella di testo denominata `txtCopione` e sostituisci la sua proprietà `Text` con il contenuto della proprietà `"txtCopione.Text"`
5. La stringa `"Ciao, mondo!"` appare all'interno della casella di testo denominata `txtMessaggio` e nella casella denominata `txtCopione` (Figura 15.4).

Quando si fa clic sul pulsante di comando Addio, ecco che cosa succede.

1. Visual Basic individua l'evento Click e nota che il mouse sta puntando a un pulsante di comando denominato `cmdAddio`.
2. Visual Basic individua rapidamente la procedura denominata `cmdAddio_Click()` e cerca ulteriori istruzioni.
3. La procedura di gestione degli eventi `cmdAddio_Click()` dice a Visual Basic: "Trova una casella di testo denominata `txtMessaggio` e sostituisci la sua proprietà `Text` con la stringa `"Addio, mondo crudele!"`".
4. Visual Basic nota la seconda istruzione che dice: "Trova una casella di testo denominata `txtCopione` e sostituisci la sua proprietà `Text` con il contenuto della proprietà `"txtMessaggio.Text"`".

5. La stringa "Addio, mondo crudele!" appare all'interno della casella di testo denominata TxtMessaggio e nella casella di testo denominata txt Copione.

Attribuire dei valori agli oggetti memorizzati in altre form

Per attribuire un valore a una proprietà di un oggetto, va utilizzato questo comando:

```
NomeOggetto.NomeProprietà = Valore
```

NomeOggetto è il nome dell'oggetto. NomeProprietà è la proprietà che si desidera modificare. Valore è il numero o la stringa che si vuole assegnare alla proprietà che agirà sull'oggetto chiamato NomeOggetto.

Volendo modificare la proprietà su una form che non è ancora stata caricata, andrà specificato il nome della form in questo modo:

```
NomeForm!NomeOggetto.NomeProprietà = Valore
```

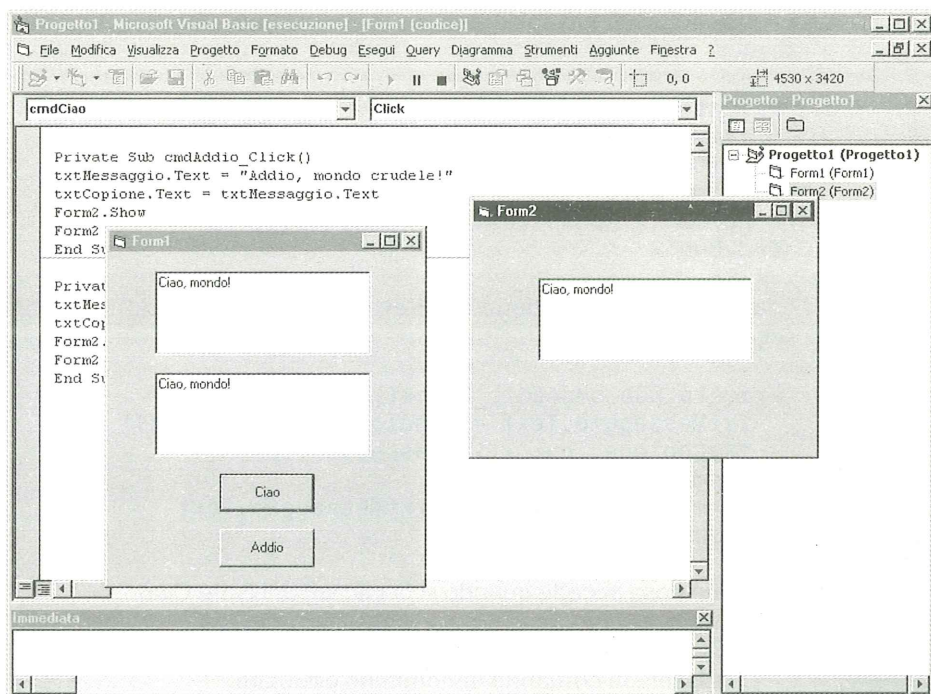
Per esempio, per modificare la proprietà Text della casella di testo txtMessaggio, si deve digitare quanto segue:

```
txtMessaggio.Text = "Ciao, mondo!"
```

Quindi, come si fa a modificare la proprietà di un oggetto memorizzato in un'altra form? La soluzione è semplice: è sufficiente specificare il nome della form nella quale è memorizzato l'oggetto.

Per esempio, nella Figura 15.5 sono illustrate due form. Nella Form 1 sono contenute due caselle di testo, denominate txtMessaggio e txtCopione, e due pulsanti di comando, denominati cmdCiao e cmdAddio. Nella Form 2 è contenuta una casella di testo con le seguenti proprietà:

Oggetto	*Proprietà*	*Valore*
Form	Name	Una seconda form
Casella di testo	Name	txtNuovo
	Text	(vuoto)

Figura 15.5 Facendo clic sul pulsante di comando Ciao, la stringa "Ciao, mondo!" viene visualizzata in tutte e tre le caselle di testo

In che modo la procedura di gestione degli eventi memorizzata nella Form 1 può modificare la proprietà Text di un oggetto memorizzato in un'altra form? È sufficiente utilizzare il seguente comando:

```
NomeForm!NomeOggetto.NomeProprietà = Valore
```

`NomeForm` specifica il nome della form che contiene l'oggetto che si desidera modificare. `NomeOggetto` è il nome dell'oggetto. `NomeProprietà` è la proprietà che si vuole modificare. `Valore` è il numero o la stringa che sarà assegnato alla proprietà.

Se volete che la casella di testo denominata `txtNuovo` visualizzi lo stesso messaggio delle caselle di testo `txtMessaggio` e `txtCopione`, dovete aggiungere il seguente comando alle procedure di gestione degli eventi `cmdCiao_Click()` e `cmdAddio_Click()`:

```
Form2!txtNuovo.Text = txtMessaggio.Text
```

La procedura di gestione degli eventi cmdCiao_Click() assomiglia a quanto segue:

```
Private Sub cmdCiao_Click()
  txtMessaggio.Text = "Ciao, mondo!"
  txtCopione.Text = txtMessaggio.Text
  Form2.Show
  Form2!txtNuovo.Text = txtMessaggio.Text
End Sub
```

e la procedura di gestione degli eventi cmdAddio_Click() assomiglia a quanto segue:

```
Private Sub cmdAddio_Click()
  txtMessaggio.Text = "Addio, mondo crudele!"
  txtCopione.Text = txtMessaggio.Text
  Form2.Show
  Form2!txtNuovo.Text = txtMessaggio.Text
End Sub
```

Ecco che cosa accade quando si fa clic sul pulsante Ciao.

1. Visual Basic individua l'evento Click e nota che il mouse sta puntando a un pulsante di comando denominato cmdCiao.
2. Visual Basic individua rapidamente la procedura denominata cmdCiao_Click() e cerca ulteriori istruzioni.
3. La procedura di gestione degli eventi cmdCiao_Click() dice a Visual Basic: "Trova una casella di testo denominata txtMessaggio" e sostituisci la sua proprietà Text con la stringa "Ciao, mondo!".
4. Visual Basic nota la seconda istruzione che dice: "Trova una casella di testo denominata txtCopione e sostituisci la sua proprietà Text con il contenuto della proprietà "txtMessaggio.Text".
5. Il comando Form2.Show dice a Visual Basic: "Trova una form denominata Form2 e visualizzala sullo schermo".
6. Il comando Form2!txtNuovo.Text = txtMessaggio.Text dice a Visual Basic: "Sulla form denominata Form2, cerca una casella di testo denominata txtNuovo e riempi la sua proprietà Text con il valore memorizzato nella casella di testo txtMessaggio".
7. La stringa "Ciao, mondo!" appare in tutte e tre le caselle di testo, ovvero txtMessaggio, txtCopione e txtNuovo. (Figura 15.5).

Quando si fa clic sul pulsante di comando Addio, ecco che cosa succede.

1. Visual Basic individua l'evento Click e nota che il mouse sta puntando a un pulsante di comando denominato cmdAddio.

2. Visual Basic individua rapidamente la procedura denominata cmdAddio_Click() e cerca ulteriori istruzioni.
3. La procedura di gestione degli eventi cmdAddio_Click() dice a Visual Basic: "Trova una casella di testo denominata txtMessaggio e sostituisci la sua proprietà Text con la stringa "Addio, mondo crudele!"".
4. Visual Basic nota la seconda istruzione che dice: "Trova una casella di testo denominata txtCopione e sostituisci la sua proprietà Text con il contenuto della proprietà "txtMessaggio.Text"".
5. Il comando Form2.Show dice a Visual Basic: "Trova una form denominata Form2 e visualizzala sullo schermo".
6. Il comando Form2!txtNuovo.Text = txtMessaggio.Text dice a Visual Basic: "Sulla form denominata Form2, cerca una casella di testo denominata txtNuovo e riempi la sua proprietà Text con il valore memorizzato nella casella di testo txtMessaggio".
7. La stringa "Addio, mondo crudele!" appare in tutte e tre le caselle di testo, ovvero txtMessaggio, txtCopione e txtNuovo.

Tipi di dati

Le variabili possono contenere numeri e stringhe, tuttavia, potreste volere che una variabile denominata NomeBattesimo contenga solo stringhe. Se una variabile NomeBattesimo dovesse contenere il numero 56, quella variabile potrebbe provocare un errore se il computer si aspettasse una stringa e ricevesse invece un numero. Per limitare il tipo di informazioni che una variabile può contenere, è necessario dichiarare che una variabile accetta soltanto un determinato tipo di dati. I tipi di dati dicono a Visual Basic: "Vedi questa variabile, può contenere soltanto stringhe oppure determinati tipi di numeri".

Ecco tre ragioni importanti per utilizzare i tipi di dati:

- ✓ per poter facilmente riconoscere il tipo di dati presente in ciascuna variabile;
- ✓ per evitare che le variabili contengano involontariamente il tipo di dati sbagliato e causino così un errore;
- ✓ per utilizzare la memoria in maniera più efficiente dato che alcuni tipi di dati (come il tipo Double) richiedono più memoria di altri (come il tipo Byte).

Se si scrive un comando BASIC che cerca di assegnare una stringa a un tipo di dati che accetta solo numeri, Visual Basic si lamenta e visualizza un messaggio di errore. Questo sistema permette di identificare eventuali errori presenti nel programma molto prima di finirlo e di distribuirlo.

Visual Basic offre dieci tipi di dati, illustrati nella Tabella 15.1.

Tabella 15.1 I tipi di dati di Visual Basic

Tipo di dati	*Accetta numeri che vanno da...*
Byte	da 0 a 255
Boolean	True o False
Currency	-922337203685477,5808 a 922337203685477,5807
Date	date dal 1 gennaio 100 al 31 dicembre 9999
Double	-1,79769313486232E308 a -4,94065645841247E-324 e 4,94065645841247E-324 a 1,79769313486232E308
Integer	-32.768 a 32.767
Long	-2.147.483.648 a 2.147.483.647
Numbers	Come Double
Single	-3,402823E38 a -1,401298E-45 e 1,401298E-45 a 3,402823E38
String	0 a 65.000 caratteri
Variant	(quando contiene numeri, come Double; quando contiene stringhe, come String).

Visual Basic offre due sistemi per dichiarare una variabile:

```
Dim MiaStringa As String
```

oppure

```
Dim MiaStringa$
```

Il primo metodo può apparire prolisso, ma è chiaro. Il secondo metodo, invece, si avvale dei cosiddetti *caratteri di dichiarazione del tipo*, che semplificano la dichiarazione delle variabili ma rendono i comandi più difficili da leggere e capire a prima vista.

Se desiderate essere chiari e non vi importa dover digitare molte parole in più come "As String" o "As Integer", adottate il primo metodo. Se invece volete risparmiare tempo e non vi interessa che il vostro codice sia leggibile, utilizzate il secondo metodo. Quella che segue è una breve tabella elencante tutti i caratteri di dichiarazione del tipo che potrete utilizzare per dichiarare le variabili.

Tipo di dati	Carattere	Esempio	Equivalente a
Currency	@	Dim Grana@	Dim Grana As Cur- rency
Double	#	Dim Media#	Dim Media As Dou- ble
Integer	%	Dim Età%	Dim Età% As Integer
Long	&	Dim Enorme&	Dim Enorme As Long
Single	!	Dim Microbo!	Dim Microbo As Sin- gle
String	$	Dim Nome$	Dim Nome As String

Dichiarare gli oggetti come tipi di dati

Per memorizzare numeri interi si utilizzi il tipo di dati Integer. Se vanno memorizzati numeri molto grandi o molto piccoli, utilizzate il tipo di dati Long. Se invece desiderate memorizzare solo numeri non inferiori a 0 e non superiori a 255, utilizzate il tipo di dati Byte.

Per memorizzare numeri con punti decimali, si utilizzi il tipo di dati Single. Se devono essere memorizzati numeri molto grandi o molto piccoli con punti decimali, si utilizzi il tipo di dati Double o Numbers.

Per memorizzare numeri che rappresentano la valuta (vale a dire soldi, con un termine non tecnico), si utilizzi il tipo di dati Currency.

Per memorizzare parole e lettere, si utilizzi il tipo di dati String.

Per memorizzare le date, si utilizzi il tipo di dati Date per memorizzare i valori True e False, utilizzate il tipo di dati Boolean.

È possibile utilizzare Variant anche per memorizzare numeri e stringhe.

Per default, Visual Basic attribuisce a tutte le variabili un tipo di dati Variant, a meno che non sia stata specificata una cosa diversa. L'unico motivo per dichiarare esplicitamente una variabile come Variant è la maggior chiarezza di lettura del codice. Per quanto riguarda Visual Basic, dichiarare una variabile come Variant è del tutto ridondante.

Per dichiarare una variabile come un particolare tipo di dati, si utilizzi questo comando:

```
Dim NomeVariabile As TipoDati
```

Per esempio, per dichiarare una variabile denominata `MioNome` come stringa, utilizzate il seguente comando:

```
Dim MioNome As String
```

Quando Visual Basic individua questa istruzione, pensa: "OK, questa è una variabile denominata `MioNome` e il programmatore l'ha definita come tipo di dati `String` in modo che possa contenere solo stringhe".

Quando Visual Basic vede l'istruzione:

```
Dim MioNome As Variant
```

pensa: "OK, questa è una variabile denominata MioNome e il programmatore l'ha definita come tipo di dati Variant in modo che possa contenere sia numeri che stringhe".

E quando Visual Basic individua l'istruzione:

```
Dim MioNome
```

pensa: "OK, questa è una variabile denominata MioNome; dato che il programmatore era troppo pigro per definire il tipo di dati, posso automaticamente supporre che questa variabile sia di tipo Variant".

Se si desidera dichiarare variabili multiple su una sola riga, è necessario farlo esplicitamente per ognuna nel seguente modo:

```
Dim MioNome As String, NomeMicia As String
```

Supponete di dichiarare le variabili in questo modo:

```
Dim MioNome, NomeMicia As String
```

Visual Basic considererebbe la variabile `MioNome` di tipo `Variant` e la variabile `NomeMicia` di tipo `String`.

Una procedura di gestione degli eventi quotidiani

Per studiare una semplice procedura di gestione degli eventi che dichiara delle variabili come tipi di dati specifici, esaminate quanto segue alla pagina successiva.

```
Private Sub Form1_Load()
Dim NomeMicia As String, Anni As Integer
  NomeMicia = "Olimpia la gatta"
  Anni = 6
End Sub
```

L'istruzione di dichiarazione della variabile `Dim NomeMicia As String, Anni As Integer` dice a Visual Basic: "Bene, crea una variabile denominata `NomeMicia` e controlla che contenga solo stringhe".

Visual Basic continua. "Poi crea una variabile denominata `Anni` e controlla che contenga soltanto numeri maggiori o uguali a -32.768 e minori o uguali a 32.767".

Il comando successivo dice a Visual Basic: "Assegna la stringa `Olimpia la gatta` alla variabile `NomeMicia`". Più rapido di un fulmine, Visual Basic controlla che la variabile `NomeMicia` contenga veramente dei valori di stringa. Fortunatamente, l'istruzione `NomeMicia As String` ha definito `NomeMicia` in modo che contenga solo valori di stringa, quindi tutto è a posto.

Infine Visual Basic dice: "Attribuisci il numero 6 alla variabile `Anni`". Visual Basic controlla velocemente che la variabile `Anni` possa contenere un numero grande quanto 6. Dato che 6 rientra nell'intervallo dichiarato (tra -32.768 e 32.767), Visual Basic accetta di buon grado questa istruzione.

Utilizzate sempre i tipi di dati più piccoli possibile. Per esempio, se sapete che la variabile `Anni` non potrà mai contenere un numero superiore a 32.767, dichiarate la variabile come Byte. Se avete bisogno di dati più grandi o più piccoli, utilizzate il tipo di dati Long. Scegliendo il tipo di dati corretto, potrete ottimizzare l'uso della memoria da parte del vostro computer.

Tipi di dati String

Nel caso si desideri limitare la lunghezza delle stringhe che una variabile può contenere, è possibile definire la lunghezza massima utilizzando il seguente comando:

```
Dim NomeVariabile As String * Size
```

Il valore di `Size` può variare da 1 a 65.500, quindi se si vuole impedire che una variabile memorizzi più di dieci caratteri, va utilizzato il seguente comando per specificare `Size` come `10`:

```
Dim NomeVariabile As String * 10
```

Per esempio, si supponga di avere dichiarato quanto segue:

```
Dim NomeBattesimo As String * 5
```

Le stringhe che seguono sarebbero valide per la variabile NomeBattesimo:

```
"12345"
"Anna"
"Carlo"
"Maria"
```

Se si prova ad assegnare alla variabile NomeBattesimo una stringa più lunga di cinque caratteri, ecco che cosa succede.

Attribuzione stringa	*Che cosa succede*
NomeBattesimo = "Giovanna"	NomeBattesimo="Giova"
NomeBattesimo = "Francesco"	NomeBattesimo="Franc"
NomeBattesimo = "Re Edoardo"	NomeBattesimo="Re Ed"

Se una stringa è più lunga della lunghezza dichiarata di una variabile, Visual Basic si limiterà a troncarla. Se non definite una lunghezza massima oppure definite la variabile come Variant, tale variabile potrà contenere fino a 65.500 caratteri.

Visibilità delle variabili

La *visibilità* di una variabile determina l'accessibilità di una variabile all'interno di un programma Visual Basic. Visual Basic vi consente di dichiarare la visibilità delle variabili in tre modi:

- ✓ locale;
- ✓ modulo;
- ✓ pubblico.

Una variabile *locale* esiste soltanto all'interno della procedura nella quale è stata creata e può essere utilizzata solo all'interno della procedura nella quale è stata dichiarata. Lo scopo di una variabile locale è isolare una variabile specifica nella singola procedura nella quale viene utilizzata, in modo che, se tale variabile contenesse un valore errato, risulterebbe semplice identificare il problema e risolverlo.

Per dichiarare una variabile locale, è necessario farlo all'interno di una procedura di gestione degli eventi nel seguente modo:

```
Private Sub Command1_Click()
  Dim NomeCompleto As String
End Sub
```

Una variabile locale può essere utilizzata solo nella procedura di gestione degli eventi nella quale viene dichiarata. Ma che cosa succede se si vuole creare una variabile condivisibile da due o più procedure di gestione degli eventi? In quel caso va creata una variabile di modulo.

Le variabili di *modulo* possono essere utilizzate soltanto dalle altre procedure memorizzate nello stesso file.

Per dichiarare una variabile di modulo, si utilizzi la procedura di seguito riportata.

1. **Aprite la finestra Codice. Aprite la finestra Codice premendo F7, selezionando Visualizza-Codice oppure facendo doppio clic su un punto qualsiasi della form (ma non sugli oggetti presenti nella form).**
2. **Fate clic nella casella di riepilogo Oggetto nella finestra Codice e selezionate (General).**

   Visual Basic visualizza (Declarations) nella casella di riepilogo Procedura.
3. **Digitate la dichiarazione della vostra variabile utilizzando il comando `Dim`, come illustrato nella Figura 15.6.**

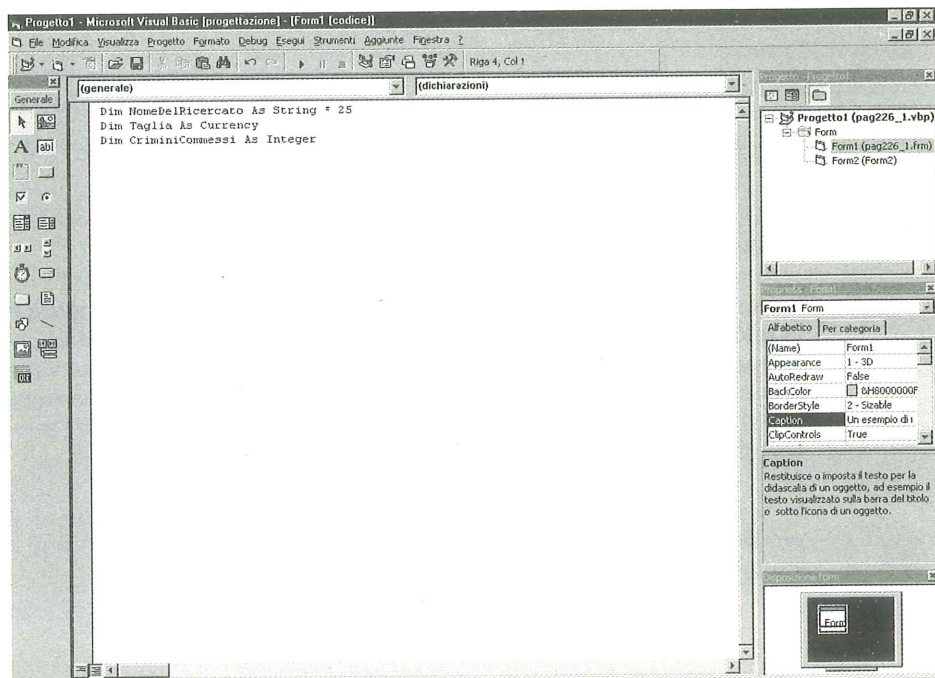



Figura 15.6 Le variabili di modulo vengono sempre dichiarate nella casella di riepilogo Oggetto (General) e nella casella di riepilogo Procedura (Declarations)

Le variabili di modulo possono essere utilizzate per condividere una variabile tra procedure memorizzate nello stesso file come lo stesso file Form (FRM). Se si vuole utilizzare una variabile che possa essere impiegata da qualsiasi procedura di gestione degli eventi, a prescindere dal file in cui è memorizzata, va creata una variabile pubblica.

Le variabili *pubbliche* possono essere le più comode da utilizzare, dato che ogni procedura di gestione degli eventi del programma Visual Basic può accedere a esse. Ma attenzione! La maggior parte dei programmatori evita di utilizzare le variabili pubbliche perché se il programma memorizza un valore errato in una variabile pubblica è necessario analizzare l'intero programma per trovare il problema.

Al contrario, se il problema riguarda una variabile di modulo, si è in grado di isolare il problema nel file in cui è stata dichiarata la variabile di modulo. In modo analogo, con una variabile locale, l'unico luogo in cui potrebbe verificarsi un pasticcio è la procedura per cui avete dichiarato la variabile locale.

Per dichiarare una variabile pubblica, si utilizzi la procedura di seguito riportata.

1. **Selezionate Progetto-Inserisci modulo per creare una file di modulo BAS.**

   La finestra di dialogo Inserisci modulo viene visualizzata.

2. **Fate clic su Modulo e poi su Apri.**

   Visual Basic visualizza la finestra Codice con (General) visualizzato nella casella di riepilogo Oggetto e (Declarations) nella casella di riepilogo Procedura.

3. **Dichiarate la variabile pubblica utilizzando il comando** `Public`**.**

   Per esempio:

```
Public NomeCompleto As String
```

Per essere certi che i programmi risultino semplici da comprendere e modificare in un momento successivo, create variabili locali, modulo o pubbliche. Se invece intendete gettare al vento qualsiasi forma di prudenza, create le variabili mano a mano che ne avete bisogno nel bel mezzo dei programmi, ma ricordate che in questo modo i vostri programmi risulteranno più difficili da leggere e capire in seguito.

Se si prevede di scrivere programmi di ampie dimensioni, vanno dichiarate sempre le variabili. All'inizio può trattarsi di un'attività disgustosa come mangiare gli spinaci, ma a lungo andare è sempre utile.

Per costringersi a dichiarare le variabili, aggiungete il seguente comando all'area Generale (come una variabile modulo) di ogni file (Form o Module) del vostro programma:

```
Option Explicit
```

Quando viene utilizzato il comando Option Explicit, Visual Basic protesta ogni volta che viene utilizzata una variabile che non sia già stata dichiarata.

```
Option Explicit
```

Capitolo 16

Rispondere all'utente

In questo capitolo

- Ottenere dati dalle caselle di testo
- Determinare quale pulsante di opzione e quale casella di controllo ha selezionato l'utente
- Recuperare dati dalle caselle di riepilogo e dalle barre di scorrimento

Un'interfaccia utente rende il vostro programma gradevole e simpatico. Purtroppo un'interfaccia utente gradevole e simpatica può essere anche priva di sostanza. Se volete che il vostro programma abbia qualcosa in più di una bella facciata, dovete assicurarvi che sappia rispondere all'utente.

Per rendere l'interfaccia utente capace di rispondere, il programma deve:

- ✓ accettare informazioni dall'interfaccia utente;
- ✓ calcolare un risultato;
- ✓ visualizzare il risultato sull'interfaccia utente.

Per esempio, quando l'utente seleziona una voce da una casella di riepilogo, il programma non sa che voce è stata selezionata. Osservando lo schermo, si potrebbe provare il desiderio di dire: "Ehi, stupido computer! Se io riesco a vedere che voce ha selezionato l'utente, perché tu non ci riesci?".

Tuttavia, quello che voi vedete sull'interfaccia utente non è quello che vede il computer; per il computer è ancora impossibile capire quale voce è stata selezionata dall'utente dalla casella di riepilogo.

Per dire al computer che cosa ha fatto un utente si deve scrivere un codice BASIC che recuperi le informazioni dall'interfaccia utente affinché il programma possa utilizzarle.

Ottenere i dati dall'interfaccia utente

Un'interfaccia utente non è altro che un sistema semplice utilizzato dagli utenti per fornire informazioni al programma. Per esempio, un'interfaccia utente può offrire una casella di riepilogo che visualizza diversi nomi di file sui quali l'utente può fare clic per selezionare quello desiderato.

Quando un utente fornisce informazioni a un programma (preme un tasto particolare, fa clic su un pulsante di comando, su una casella di riepilogo e così via), Visual Basic memorizza queste informazioni nella proprietà dell'oggetto. Da questo momento in poi il programma potrà utilizzare le informazioni ricevute.

Nella Figura 16.1 sono illustrati i diversi modi che un utente può utilizzare per fornire informazioni al programma facendo clic su un oggetto oppure digitando dei dati.

Questi i nove tipi di oggetti di base in grado di accettare dati dall'interfaccia utente:

- ✓ caselle di controllo
- ✓ pulsanti di opzione
- ✓ caselle combinate
- ✓ caselle di riepilogo
- ✓ caselle di riepilogo unità

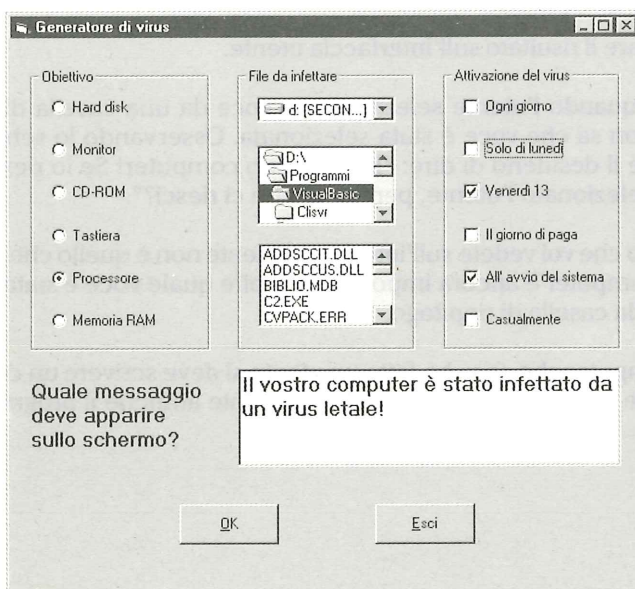



Figura 16.1 I diversi metodi che un utente può utilizzare per fornire informazioni al programma

- caselle di riepilogo directory
- caselle di riepilogo file
- barre di scorrimento orizzontali e verticali
- caselle di testo.

Trovare le informazioni contenute in un oggetto

Per trovare le informazioni memorizzate in un oggetto, si devono conoscere:

- il nome della form su cui si trova l'oggetto; (come txt ScriviQui)
- il nome dell'oggetto; (come la proprietà Text)
- la proprietà dell'oggetto che contiene le informazioni (come frmMain).

Quando l'utente inserisce informazioni in un oggetto (come un testo in una casella di testo), Visual Basic memorizza i dati in una delle *proprietà* dell'oggetto. Oggetti diversi memorizzano le informazioni fornite dall'utente in proprietà diverse; per esempio, una casella di testo memorizza le informazioni nella sua proprietà Text, mentre una casella di controllo memorizza le informazioni nella proprietà Value.

Sapere quale proprietà si desidera non è sufficiente: è necessario conoscere anche il nome dell'oggetto che contiene la proprietà desiderata. Inoltre, dato che gli oggetti su diverse form possono avere lo stesso nome, è necessario conoscere anche il nome della form.

La combinazione di nome della form, nome dell'oggetto e proprietà definisce la posizione specifica delle informazioni immesse dall'utente. Pensate agli oggetti come a caselle della posta: per recuperare una lettera dovete conoscere la provincia (nome della form), la città (nome dell'oggetto) e l'indirizzo (proprietà).

Quindi, se si vogliono recuperare informazioni da un oggetto, va utilizzata la seguente combinazione:

```
NomeForm!NomeOggetto.Proprietà
```

Per esempio, si potrebbe voler recuperare le informazioni da una form denominata `frmAttacco`, che contiene una casella di testo denominata `txtAvvertimento`, che memorizza i dati nella proprietà `Text` nel seguente modo:

```
frmAttacco!txtAvvertimento.Text
```

Ogni volta che Visual Basic vede questo codice, dice automaticamente a se stesso: “Benissimo, troviamo la form che si chiama `frmAttacco`, cerchiamo l’oggetto denominato `txtAvvertimento` e individuiamo il valore memorizzato nella proprietà `Text`. Ah, eccolo”.

Se non viene inserito il nome della form, Visual Basic cerca gli oggetti soltanto sulla form visualizzata in quel momento. Se è qui che si vuole cercare, è possibile abbreviare l’istruzione scrivendo solo quanto segue:

```
NomeOggetto.Proprietà
```

Recuperare i dati dalle caselle di testo

Quando l’utente digita qualcosa in una casella di testo, Visual Basic memorizza le informazioni nella proprietà Text. Digitare dati in una casella di testo equivale ad attribuire un valore alla proprietà Text della casella di testo (Figura 16.2).

Se l’utente digita “Saluti da Marte” in una casella di testo denominata `txtSegreto`, questa azione corrisponde al seguente codice BASIC:

```
txtSegreto.Text = "Saluti da Marte!"
```

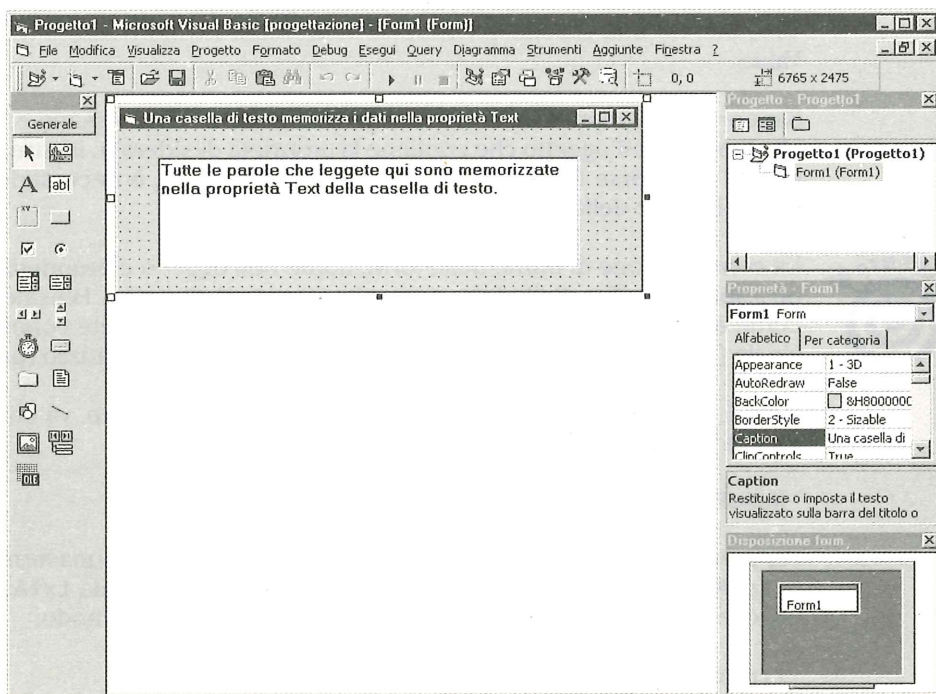



Figura 16.2 La memorizzazione in una casella di testo

Determinare il pulsante di opzione selezionato dall'utente

Un pulsante di opzione può contenere solo due valori memorizzati nella proprietà Value:

- True (selezionato)
- False (non selezionato).

Un pulsante di opzione non selezionato ha la proprietà Value impostata a False. Facendo clic su un pulsante di opzione non selezionato, si modifica la sua proprietà Value e la si imposta a True.

Un pulsante di opzione selezionato ha la proprietà Value impostata a True. Facendo clic su un pulsante di opzione selezionato, si modifica la sua proprietà Value e la si imposta a False.

Per determinare quale pulsante di opzione è stato selezionato dall'utente, va controllata la proprietà Value di tutti i pulsanti di opzione. Per esempio, se l'utente ha scelto un pulsante di opzione non selezionato denominato `optStazione`, questa selezione equivale al seguente codice BASIC:

```
optStazione.Value = True
```

Se l'utente seleziona un pulsante di opzione diverso (da quello chiamato optStazione), questa azione equivale al seguente codice BASIC:

```
optStazione.Value = False
```

Determinare la casella di controllo selezionata dall'utente

Una casella di controllo ha la possibilità di contenere solo uno dei tre seguenti valori nella proprietà Value:

- 0 non selezionata
- 1 selezionata
- 2 ombreggiata.

Nella Figura 16.3 sono illustrati i possibili valori delle caselle di controllo. Una casella di controllo non selezionata ha la propria proprietà Value impostata a 0. Facendo clic su una casella di controllo non selezionata, si modifica la sua proprietà Value e la si imposta a 1.

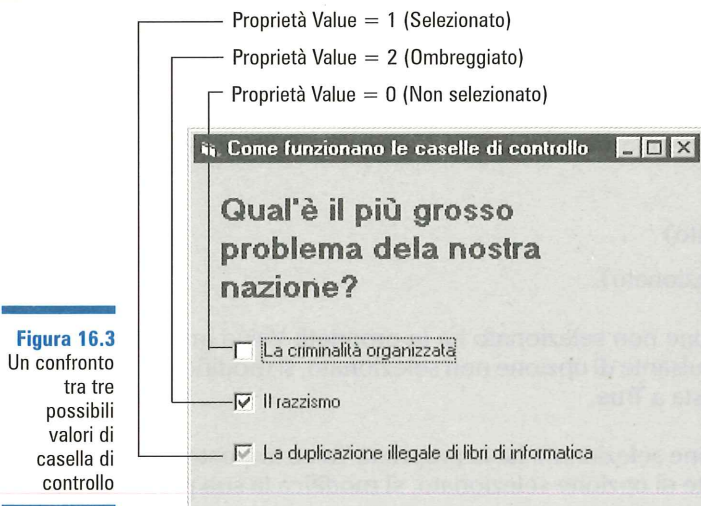



Figura 16.3 Un confronto tra tre possibili valori di casella di controllo

Una casella di controllo selezionata ha la propria proprietà Value impostata a 1. Facendo clic su una casella di controllo selezionata, si modifica la sua proprietà Value e la si imposta a 0.

Una casella di controllo ombreggiata ha la sua proprietà Value impostata a 2. Facendo clic su una casella di controllo ombreggiata, si modifica la sua proprietà Value e la si imposta a 0.

Si noti che è possibile fare clic su una casella di controllo ombreggiata; al contrario di una casella di controllo non attivata visualizzata anch'essa in grigio.

Una casella di controllo selezionata sta a significare ovviamente che il comando rappresentato dalla casella di controllo è stato selezionato. Analogamente, una casella di controllo non selezionata indica che il comando rappresentato dalla casella di controllo non è stato selezionato. Una casella di controllo ombreggiata può invece essere utilizzata per mostrare all'utente che il comando rappresenta un'eccezione o una funzione speciale.

Sono disponibili solo due sistemi per impostare a 2 la proprietà Value di una casella di controllo e fare in modo che appaia ombreggiata; il primo sistema consiste nel modificare la proprietà Value della finestra Property durante la progettazione del programma, mentre il secondo consiste nell'utilizzo del seguente codice BASIC, che imposta a 2 la proprietà `Value` di una casella di controllo denominata `chkGrassetto`:

```
chkGrassetto.Value = 2
```

Per determinare quale casella di controllo è stata selezionata dall'utente, va esaminata la proprietà Value di tutte le caselle di controllo. Per esempio, se una casella di controllo denominata `chkGrassetto` risulta vuota, il codice BASIC equivalente è:

```
chkGrassetto.Value = 0
```

Se la casella di controllo è selezionata, il codice BASIC equivalente è:

```
chkGrassetto.Value = 1
```

Se la casella di controllo è ombreggiata, il codice BASIC equivalente è:

```
chkGrassetto.Value = 2
```

Recuperare i dati dalle caselle di riepilogo unità, directory e file

Quando l'utente compie una selezione da una casella di riepilogo unità (drive), Visual Basic memorizza la selezione sotto forma di una stringa nella proprietà Drive della casella di riepilogo Unità. Se per esempio un utente fa clic su `c:` in una casella di riepilogo Unità denominata `drvQualeUnità`, si può immaginare che sia stato eseguito un comando simile al seguente:

```
drvQualeUnità.Drive = "c:"
```

Quando un utente esegue una selezione dalla casella di riepilogo Directory, Visual Basic memorizza la selezione sotto forma di stringa nella proprietà Path della casella di riepilogo Directory.

Se per esempio l'utente fa clic su `c:\dos` nella casella di riepilogo Directory denominata `dirCartella`, si può immaginare che sia stato eseguito un comando simile al seguente:

```
dirCartella.Path = "c:\dos"
```

Quando un utente esegue una selezione da una casella di riepilogo File, Visual Basic memorizza il nome del file sotto forma di una stringa nella proprietà FileName della casella di riepilogo File.

Se per esempio l'utente fa clic su `autoexec.bat` nella casella di riepilogo File denominata `filCercaFile`, si può immaginare che sia stato eseguito un comando simile al seguente:

```
filCercaFile.Filename = "c:\autoexec.bat"
```

Determinare la scelta effettuata in una casella combinata

Quando l'utente seleziona o digita una voce in una casella combinata, Visual Basic memorizza tali informazioni nella proprietà Text della casella combinata (Figura 16.4.).

Il testo visualizzato da una casella combinata è memorizzato nella proprietà List di detta casella. Quando l'utente digita o seleziona una voce in una casella combinata, la scelta dell'utente viene memorizzata nella proprietà Text della casella.

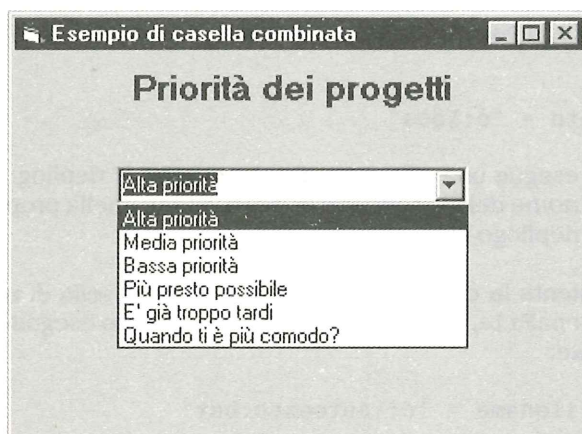
Se per esempio l'utente digita o seleziona una voce denominata `"Alta priorità"` in una casella combinata chiamata `cboPriorità`, si può immaginare che sia stato eseguito un comando simile al seguente:

```
cboPriorità.Text = "Alta priorità"
```

Ottenere dei valori dalle barre di scorrimento orizzontali e verticali

Le barre di scorrimento che non fanno parte di una casella di testo, di una casella di riepilogo o di una casella combinata rappresentano un numero che viene memorizzato nella proprietà Value della barra di scorrimento.

Il valore che una barra di scorrimento può rappresentare è determinato dalle proprietà Min e Max della barra di scorrimento stessa; il valore più basso possibile è -32.768, mentre il valore massimo possibile è 32.767.

Figura 16.4 La proprietà Text di questa casella combinata contiene la stringa "Alta Priorità"

Mettete alla prova le vostre nuove conoscenze

1. Che cosa fa il seguente comando BASIC?

```
CheCosaE = chkGrassetto.Value
```

a. Tenta di identificare gli UFO che si chiamano chkGrassetto.Value.

2. Il comando mette in dubbio la necessità di qualcosa che si chiami chkGrassetto.Value.

c. Il comando fa chiedere al computer: "Che cosa vuoi da me? Dimmelo e poi lasciami stare. Oggi ho la luna storta".

d. Il comando recupera un valore memorizzato nella proprietà `Value` di una casella di controllo denominata `chkGrassetto` e inserisce questo numero in una variabile denominata `CheCosaE`. Se la casella di controllo `chkGrassetto` fosse stata selezionata, il valore di `CheCosaE` sarebbe stato True.

2. Esaminate il seguente comando e spiegate che cosa fa.

```
FrmFoglioDati!txtMessaggio.Text
= "Attenzione!"
```

a. Il comando dice a Visual Basic: "OK, cerca una form denominata `frmFoglioDati` e su questa form trova una casella di testo denominato `txtMessaggio`. Quando l'avrai trovata, inserisci la stringa `"Attenzione"` nella sua proprietà `Text`.

b. Il comando dice a tutti che la persona che ha scritto questo comando molto probabilmente aveva un'ottima ragione per non essere presente e aiutarvi a interpretare il comando.

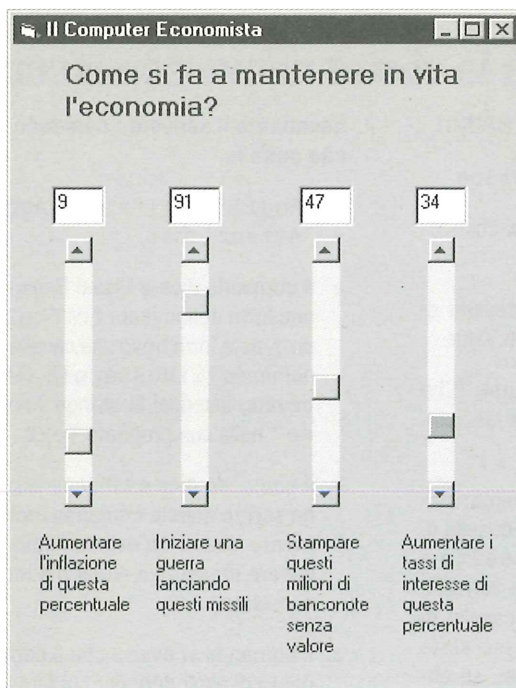
c. Il comando vi avvisa che il computer è sul punto di esplodere per cui fareste meglio a mettervi immediatamente al riparo.

Le barre di scorrimento permettono agli utenti di rappresentare un valore in modo visivo invece di dover digitare un numero mediante tastiera. In linea di principio, si dovrebbe utilizzare la barra di scorrimento insieme a un'etichetta che mostri il valore effettivo della barra; in questo modo, quando gli utenti sposteranno la casella di scorrimento nella barra di scorrimento, potranno anche seguire la variazione del valore, come illustrato nella Figura 16.5.

Sebbene le barre di scorrimento possano rappresentare una serie di valori, alla fine il programma dovrà leggere un singolo valore dalla barra di scorrimento.

Se per esempio si vuole recuperare il valore di una barra di scorrimento orizzontale denominata `hsbSensibilità`, occorre impostare una variabile uguale alla proprietà Value della barra di scorrimento. Quindi, se avete una variabile chiamata SensibilitàMouse e una barra di scorrimento chiamata hsbSensibilità, potrete utilizzare il seguente comando:

```
SensibilitàMouse = hsbSensibilità.Value
```

Figura 16.5 Come utilizzare un'etichetta per visualizzare il valore di una barra di scorrimento

Recuperare i dati da una casella di riepilogo

Un utente può selezionare una o più voci da una casella di riepilogo a seconda di come è impostata la proprietà MultiSelect. Se la proprietà MultiSelect è impostata a 0 (il valore di default) e l'utente seleziona una voce, la casella di riepilogo memorizza la voce nella proprietà Text.

Se l'utente seleziona la voce denominata `"Cibo per gatti"` in una casella di riepilogo denominata `lstAlimentari`, si può immaginare che sia stato eseguito un comando simile al seguente:

```
lstAlimentari.text = "Cibo per gatti"
```

Se la proprietà MultiSelect di una casella di riepilogo è 1 o 2, gli utenti possono selezionare due o più voci tenendo premuto il tasto MAIUSC e facendo clic su ognuna delle voci desiderate. Dato che la proprietà Text di una casella di riepilogo può memorizzare solo una stringa per volta, è impossibile che tutte le selezioni effettuate da un utente siano accessibili mediante la lettura della proprietà Text.

Per recuperare i dati da una casella di riepilogo che consente selezioni multiple, si deve utilizzare la procedura di seguito riportata.

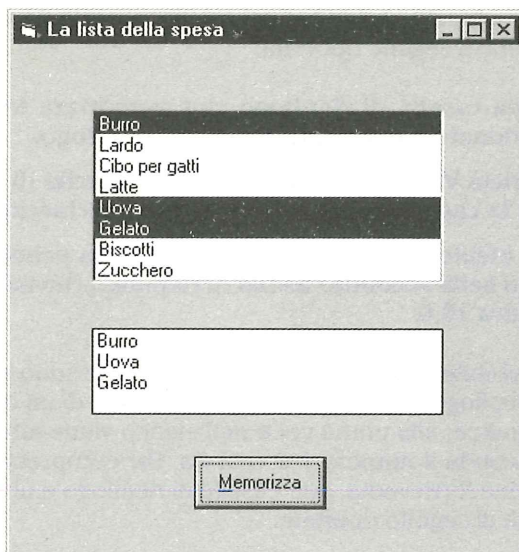
1. **Create una seconda casella di riepilogo che memorizza temporaneamente le voci selezionate dalla prima casella di riepilogo.**
2. **Impostate la proprietà Visibile di questa seconda casella di riepilogo a False in modo che la casella risulti invisibile sull'interfaccia utente.**
3. **Ogni volta che un utente seleziona un elemento della prima casella di riepilogo, copiatelo nella seconda casella di riepilogo (invisibile), come illustrato nella Figura 16.6.**

La casella di riepilogo invisibile contiene solo gli elementi che sono stati selezionati dall'altra casella di riepilogo, nella proprietà List. Alle voci di un elenco viene attribuito un numero di indice; alla prima voce dell'elenco viene attribuito il numero di indice 0, alla seconda il numero 1 e così via. Per comprendere meglio questo concetto, create una form vuota, due caselle di riepilogo e un pulsante di comando con le proprietà di seguito riportate.

Se non avete voglia di digitare il seguente programma, caricate LISTBOXES.VBP dal CD-ROM.

Oggetto	*Proprietà*	*Valore*
Prima casella di riepilogo	Name	lstScelta
	MultiSelect	1 - Simple
Seconda casella di riepilogo	Name	lstTemp
	Visible	True
Pulsante di comando	Name	cmdMemorizza
	Caption	Memorizza

Se la proprietà Visible della seconda casella di riepilogo è False, la casella è (ovviamente) invisibile e non si riesce a vedere che cosa sta succedendo. Per questo motivo, quanto proverete la procedura, impostate la proprietà Visible della seconda casella di riepilogo a True, in modo da poter controllare che cosa succede. Quindi, dopo che avrete "imparato", modificate nuovamente la proprietà Visibile e impostatela a False.

Figura 16.6 Come recuperare le voci da una casella di riepilogo utilizzando una seconda casella di riepilogo (invisibile)

La procedura seguente memorizza delle voci nella casella di riepilogo `lstScelta`:

```
Private Sub Form_Load()
  lstScelta.AddItem "Burro"
  lstScelta.AddItem "Lardo"
  lstScelta.AddItem "Cibo per gatti"
  lstScelta.AddItem "Latte"
  lstScelta.AddItem "Uova"
  lstScelta.AddItem "Gelato"
  lstScelta.AddItem "Biscotti"
  lstScelta.AddItem "Zucchero"
End Sub
```

Una procedura di gestione degli eventi scritta per recuperare diverse voci selezionate dalla casella di riepilogo `lstScelta` potrebbe assomigliare a quanto segue:

```
Private Sub cmdMemorizza_Click()
  Dim I
  lstTemp.Clear
  For I = 0 To lstScelta.ListCount -1
     If lstScelta.Selected(I) Then
     lstTemp.AddItem lstScelta.List(I)
     End If
  Next I
End Sub
```

Ecco in che modo Visual Basic risponde al codice.

1. La prima riga dice a Visual Basic: "Segui queste istruzioni ogni volta che l'utente fa clic sul pulsante di comando denominato `cmdMemorizza`".
2. La seconda riga dice: "Dichiara una variabile denominata I e presupponi che il tipo di dati sia Variant".
3. La terza riga dice: "Trova una casella di riepilogo denominata `lstTemp` e svuotala completamente".
4. La quarta riga dice: "Imposta a 0 il valore di `I` e continua il conteggio con incrementi di 1 fino a quando il valore di `I` corrisponderà al numero delle voci visualizzate nella casella di riepilogo `lstScelta`".
5. La quinta riga dice: "Se l'utente ha selezionato una voce visualizzata nella casella di riepilogo `lstScelta`, procedi secondo le istruzioni che seguono, se no passa alla riga 8".
6. La sesta riga dice: "Copia la voce selezionata nella casella di riepilogo `lstScelta` e inserisci la voce copiata nella casella di riepilogo `lstTemp`".
7. La settima riga dice: "Questa è la fine dellle istruzioni da seguire se la voce è selezionata".
8. L'ottava riga dice: "Aggiungi 1 al valore di `I` e ritorna alla quinta riga".
9. La nona riga dice: "Questa è la fine delle istruzioni da seguire quando l'utente fa clic sul pulsante di comando denominato `cmdMemorizza`".

Queste istruzioni non fanno altro che copiare tutte le voci selezionate dalla prima casella di riepilogo in una seconda casella di riepilogo che è invisibile.

Ogni voce contenuta in una casella di riepilogo (ma anche in una casella combinata) viene identificata da un numero di indice. La voce in cima all'elenco possiede il numero di indice 0, la seconda ha il numero di indice 1 e così via.

Per ottenere le voci memorizzate nella casella di riepilogo `lstTemp`, vanno utilizzati i numeri di indice; il comando seguente recupera la voce memorizzata in cima alla casella di riepilogo `lstTemp`:

```
lstTemp.List(0)
```

Quest'altro comando recupera la seconda voce nella casella di riepilogo `lstTemp`:

```
lstTemp.List(1)
```

Ogni incremento del numero di indice fa in modo che vengano recuperate voci sempre più lontane dall'inizio dell'elenco.

Capitolo 17

Matematica: operatori aritmetici, logici e di confronto

In questo capitolo

- Sommare, sottrarre, moltiplicare e dividere i numeri
- Utilizzare gli operatori `Not`, `And`, `Or` o `Xor`
- Confrontare numeri e stringhe
- Confrontare stringhe e operatori

Una volta ottenuti i dati dall'utente (come un numero o una stringa), il programma passa alla loro elaborazione. Se il vostro programma emulasse un impiegato sovraffaticato, probabilmente si limiterebbe a perdere i dati e a dare la colpa a qualcun altro; con ogni probabilità, però, vorrete, invece, che calcoli qualche risultato utilizzando i dati ricevuti dall'utente.

Per calcolare un risultato, il programma riceverà i dati dell'interfaccia utente e quindi dovrà in qualche modo cambiarli, modificarli, mutilarli o macinarli. Qualsiasi modifica richiede un'operazione, ecco perché i comandi speciali utilizzati per gestire i dati vengono detti *operatori*.

Visual Basic fornisce tre tipi di operatori.

- ✓ Aritmetici.
- ✓ Logici.
- ✓ Di confronto.

Gli operatori aritmetici

Gli operatori aritmetici sono essenzialmente quelli che trasformano il vostro computer da 4 milioni in una calcolatrice tascabile da 10.000 lire e vi permettono di sommare, sottrarre, moltiplicare e dividere numeri o variabili che rappresentano numeri. Nella Tabella 17.1 sono riportati gli operatori aritmetici più comuni.

Tabella 17.1 Operatori aritmetici

Operatore	*Funzione*
+	Somma due numeri.
-	Sottrae due numeri.
*	Moltiplica due numeri.
/	Divide due numeri e restituisce un numero a virgola mobile (decimale), come 3,14, 16,2 oppure 392,2398.
\	Divide due numeri e restituisce un numero intero, come 8, 16.012 oppure 25.
`Mod (modulo)`	Divide due numeri e restituisce solo il resto.
^	Eleva un numero alla potenza.
&	Somma (unisce) due stringhe.

Sommare due numeri utilizzando l'operatore +

Per sommare due numeri utilizzate l'operatore (+), come illustrato nel seguente esempio:

```
X = 10
Y = 15.4
Somma = X + Y
```

Nel caso questi tre comandi BASIC vi confondano, ecco come funzionano.

1. Il primo comando dice: "Crea una variabile denominata `X` e imposta il suo valore a 10".
2. Il secondo comando dice: "Crea una variabile denominata `Y` e imposta il suo valore a 15,4".
3. Il terzo comando dice: "Crea una variabile denominata `Sum` e imposta il suo valore al valore di `X` più il valore di `Y`". In questo caso, il valore di `Somma` è uguale a `10 + 15.4`, ovvero 25.4.

Sottrarre due numeri con l'operatore -

Per sottrarre due numeri utilizzate l'operatore (-), come illustrato nell'esempio a pagina successiva.

```
Reddito = 2000
Tasse = 1500
Reddito_Effettivo = Reddito - Tasse
```

Ecco come Visual Basic interpreta questi tre comandi BASIC.

1. Il primo comando dice: "Crea una variabile denominata `Reddito` e imposta il suo valore a `2000`".
2. Il secondo comando dice: "Crea una variabile denominata `Tasse` e imposta il suo valore a `1500`".
3. Il terzo comando dice: "Crea una variabile denominata `Reddito_Effettivo` e imposta il suo valore al valore del `Reddito` meno il valore delle `Tasse`". In questo caso, il valore di `Reddito_Effettivo` è uguale a `2000 - 1500`, ovvero 500.

Rendere negativi dei numeri con l'operatore -

L'operatore -, utilizzato da solo, può trasformare un numero positivo in un numero negativo e viceversa. Per rendere negativo un numero, inserite l'operatore - di fronte a qualsiasi numero o variabile, come illustrato nel seguente esempio:

```
Importo = 250
Saldo = - Importo
```

Ecco in che modo il cervellino di Visual Basic interpreta questi comandi BASIC.

1. Il primo comando dice: "Crea una variabile denominata `Importo` e imposta il suo valore a `250`".
2. Il secondo comando dice: "Crea una variabile denominata `Saldo` e imposta il suo valore al valore negativo di `Importo`". In questo caso, il valore di `Saldo` è -250.

Moltiplicare due numeri

Per moltiplicare due numeri utilizzate l'operatore *, come illustrato nel seguente esempio.

```
Ore = 25
Paga = 5.75
Stipendio: Ore * Paga
```

Ecco come Visual Basic interpreta questi tre comandi BASIC.

1. Il primo comando dice: "Crea una variabile denominata `Ore` e imposta il suo valore a `25`".
2. Il secondo comando dice: "Crea una variabile denominata `Paga` e imposta il suo valore a `5.75`".
3. Il terzo comando dice: "Crea una variabile denominata `Stipendio` e imposta il suo valore a un valore equivalente a `Ore` moltiplicato per quello di `Paga`". In questo caso, il valore di `Stipendio` è uguale a `25 * 5.75`, ovvero 143,75.

Dividere due numeri con l'operatore /

Per dividere due numeri e calcolare un numero a virgola mobile (decimale) utilizzate l'operatore / (barra obliqua), come illustrato nel seguente esempio:

```
PartiteVinte = 104
TotalePartite = 162
PercentualeVincita = PartiteVinte / TotalePartite
```

Ecco come Visual Basic interpreta questi tre comandi BASIC.

1. Il primo comando dice: "Crea una variabile denominata `PartiteVinte` e imposta il suo valore a `104`".
2. Il secondo comando dice: "Crea una variabile denominata `TotalePartite` e imposta il suo valore a `162`".
3. Il terzo comando dice: "Crea una variabile denominata `PercentualeVincita` e imposta il suo valore al valore di `PartiteVinte` diviso il valore di `TotalePartite`". In questo caso, il valore di `PercentualeVincita` è uguale a `104 / 162`, ovvero 0,6419753.

Dividere due numeri con l'operatore \

Per dividere due numeri e calcolare un valore intero, utilizzate l'operatore \ (barra obliqua inversa), come illustrato nel seguente esempio:

```
CapacitàBotti = 72
VinoProdotto = 1900
BottiIntere = VinoProdotto \ CapacitàBotte
```

In che modo Visual Basic interpreta questi tre comandi BASIC? Siamo felici che lo abbiate chiesto. Ecco qua.

1. Il primo comando dice: "Crea una variabile denominata `CapacitàBotte` e imposta il suo valore a `72`".

2. Il secondo comando dice: "Crea una variabile denominata `VinoProdotto` e imposta il suo valore a `1900`".

3. Il terzo comando dice: "Crea una variabile denominata `BottiIntere` e imposta il suo valore al valore di `VinoProdotto` diviso il valore di `CapacitàBotte`". In questo caso il valore di `BottiIntere` è uguale a `1900 \ 72`, ovvero 26.

La divisione di due numeri produce spesso dei valori a virgola mobile (decimali), quindi in che modo Visual Basic può gestire l'arrotondamento? Prendete in considerazione il seguente esempio:

```
Operando1 = 2.5
Operando2 = 1.5
Risultato = Operando1 \ Operando2
```

Prima che Visual Basic esegua un calcolo utilizzando l'operatore \, gli operandi vengono arrotondati al più vicino numero intero. (Se un operando si trova a metà tra due numeri interi, come per esempio 2,5 o 1,5, l'operando verrà arrotondato). In questo esempio, `Operando1` viene arrotondato a 3 e `Operando2` viene arrotondato a 2; di conseguenza, `Risultato = 3 \ 2`, ovvero 1,5. Dato che l'operatore \ deve fornire come risultato un valore intero, il valore di `Risultato` viene arrotondato a 1.

Dividere con Mod (modulo)

Per dividere due numeri e calcolare il resto, utilizzate l'operatore `Mod`, come illustrato nel seguente esempio:

```
CapacitàBotte = 72
BottiglieNellaBotte = 1900
BottiglieAvanzo = VinoProdotto Mod CapacitàBotte
```

Per i più curiosi, ecco come Visual Basic interpreta questi tre comandi BASIC.

1. Il primo comando dice: "Crea una variabile denominata `CapacitàBotte` e imposta il suo valore a `72`".

2. Il secondo comando dice: "Crea una variabile denominata `VinoProdotto` e imposta il suo valore a `1900`".

3. Il terzo comando dice: "Crea una variabile denominata `BottiglieAvanzo` e imposta il suo valore sul resto del valore di `VinoProdotto` diviso il valore di `CapacitàBotte`". In questo caso, il valore di `BottiglieAvanzo` è uguale a `1900 Mod 72`, ovvero 28.

Calcolare l'n-esima potenza di un numero con l'operatore ^

L'n-esima potenza di un numero rappresenta la moltiplicazione di un numero per se stesso, per un certo numero di volte. Per esempio, la moltiplicazione del numero 2 per se stesso per quattro volte viene indicata con 2^4 cioè 2*2*2*2.

Dato che non è possibile digitare 2^4 e digitare 2*2*2*2 è piuttosto scomodo, Visual Basic offre l'operatore ^ (caret), che si utilizza come illustrato nel seguente esempio:

```
2 ^ 4
```

Sommare (concatenare) due stringhe con l'operatore &

Sommare o concatenare due stringhe significa unirle; per questa operazione utilizzate l'operatore & ("e" commerciale), come illustrato nel seguente esempio:

```
NomeBattesimo = "Gianni "
Cognome = "Bronco"
NomeCompleto = NomeBattesimo & Cognome
```

Quando concatenate le stringhe, lasciate sempre uno spazio tra una stringa e l'altra, altrimenti Visual Basic le unirà formando una parola sola, come "GianniBronco".

Ecco come Visual Basic interpreta questi tre comandi BASIC.

1. Il primo comando dice: "Crea una variabile denominata `NomeBattesimo` e imposta il suo valore a `"Gianni "` (notate lo spazio alla fine)".
2. Il secondo comando dice: "Crea una variabile denominata `Cognome` e imposta il suo valore a `"Bronco"`".
3. Il terzo comando dice: "Crea una variabile denominata `NomeCompleto` e imposta il suo valore al valore di `NomeBattesimo` più il valore di `Cognome`". In questo caso, il valore di `NomeCompleto` è `"Gianni " & "Bronco"`, ovvero "Gianni Bronco".

Per concatenare le stringhe, oltre alla e commerciale (&), potete utilizzare anche il segno più (+). Tuttavia, in genere è meglio optare per la &, dato che il + viene utilizzato anche per le addizioni numeriche e si rischia di creare confusione.

Operatori logici

Gli operatori *logici* gestiscono i valori True e False. Visual Basic rappresenta un valore True come -1 e un valore False come 0. Nella Tabella 17.2 sono riportati gli operatori logici più comuni.

Tabella 17.2 Operatori logici

Operatore	*Come utilizzarlo*
`And`	Variabile1 And Variabile2
`Or`	Variabile1 Or Variabile2
`Xor`	Variabile1 Xor Variabile2
`Not`	Not variabile

Utilizzare l'operatore Not

L'operatore `Not` modifica un valore True in un valore False e un valore False in un valore True, come nel seguente esempio.

Nome della variabile	*Valore*
`Un_altro_libro_di_informatica`	True
`Not Un_altro_libro_di_informatica`	False

Per maggiore chiarezza, i programmatori amano utilizzare le parentesi; utilizzando le parentesi nell'esempio precedente si otterrebbe questo risultato:

```
Not(Un_altro_libro_di_informatica)
```

Utilizzare l'operatore And

L'operatore And confronta i valori True e False di due variabili e calcola un nuovo valore True o False. Questo permette al vostro programma di prendere delle decisioni, come viene illustrato nel seguente esempio:

```
CacciaGatto = GattoPresente And GattoMonello
```

Quando la variabile `CacciaGatto` è True e quando invece False? Dipende ovviamente dai valori delle variabili `GattoPresente` e `GattoMonello`.

CacciaGatto	GattoPresente	GattoMonello
True	True	True
False	False	False
False	True	False
False	False	True

L'operatore And restituisce un valore True soltanto se sia `GattoPresente` sia `GattoMonello` sono impostati a True.

Utilizzare l'operatore Or

In modo analogo all'operatore And, l'operatore `Or` esegue un confronto tra i valori True e False di due variabili e calcola un nuovo valore True o False. Questo permette al programma di prendere decisioni, come viene illustrato nel seguente esempio:

```
OziaACasa = PartitaTV Or TempoBrutto
```

Quando la variabile `OziaACasa` è True e quando invece è False? La relazione tra il risultato e le variabili `PartitaTV` e `TempoBrutto` è la seguente.

OziaACasa	PartitaTV	TempoBrutto
True	True	True
True	False	True
True	True	False
False	False	False

L'operatore `Or` restituisce un valore False quando sia `PartitaTV` sia `TempoBrutto` sono False.

Utilizzare l'operatore Xor

Come per gli operatori And e `Or`, l'operatore `Xor` esegue un confronto tra i valori True e False di due variabili e calcola un nuovo valore True o False. Questo permette al vostro programma di prendere delle decisioni, come viene illustrato nel seguente esempio:

```
RamanzinaAlCapo = CapoPresente Xor AlLavoro
```

Quando la variabile `RamanzinaAlCapo` è True e quando invece è False? La relazione tra il risultato e le variabili `CapoPresente` e `AlLavoro` è la seguente.

`RamanzinaAlCapo`	`CapoPresente`	`AlLavoro`
True	True	False
True	False	True
False	True	True
False	False	False

L'operatore `Xor` restituisce un valore False quando se sia `CapoPresente` sia `AlLavoro` sono contemporaneamente True oppure False.

Operatori di confronto

Gli operatori *di confronto* eseguono un confronto tra due numeri o stringhe per vedere se sono uguali, diverse, maggiori o minori l'una rispetto all'altra. Nella Tabella 17.3 sono riportati gli operatori di confronto più comuni.

Tabella 17.3 Operatori di confronto

Operatore	*Significato*
<	Minore di
<=	Minore o uguale a
>	Maggiore di
>=	Maggiore o uguale a
=	Uguale a
<>	Diverso da

Confrontare numeri e stringhe

Come mostrato nell'esempio seguente, gli operatori di confronto eseguono un confronto tra i valori di numeri e stringhe per dare come risultato un valore True o False.

```
Età = 18
EtàMinima = 21
esso = (Età >= EtàMinima)
```

Ecco come Visual Basic interpreta questi tre comandi BASIC.

1. Il primo comando dice: "Crea una variabile denominata `Età` e imposta il suo valore a `18`".
2. Il secondo comando dice: "Crea una variabile denominata `EtàMinima` e imposta il suo valore a `21`".
3. Il terzo comando dice: "Confronta il valore di `Età` e controlla se è superiore o uguale al valore di `EtàMinima`. Se così fosse, crea una variabile denominata `Permesso` e imposta il suo valore su True. In caso contrario, imposta il valore di `Permesso` su False".

Confrontare i numeri è piuttosto semplice ma confrontare le stringhe può essere più complesso. Quando vengono confrontate le stringhe, Visual Basic calcola il valore di ciascuna lettera in codice di carattere ANSI.

I codici di carattere ANSI

Al livello più semplice, i computer comprendono soltanto due numeri: zero e uno. Tutti i numeri possono essere rappresentati da una serie di zero e di uno; questi numeri vengono definiti *binari*.

Dato che i computer capiscono solo i numeri e non le lettere, gli esseri umani hanno creato un semplice sistema in cui certi numeri rappresentano particolari lettere, simboli di punteggiatura e caratteri.

Per esempio, il numero 97 rappresenta la lettera *a*, il numero 65 rappresenta la lettera *A* e il numero 33 rappresenta un punto esclamativo (*!*).

Per garantire che tutti i computer utilizzino gli stessi numeri per rappresentare le stesse lettere e gli stessi simboli di punteggiatura, l'ANSI (American National Standard Institute) ha definito un insieme di caratteri ANSI, che specifica quale numero dovrà rappresentare una particolare lettera o un particolare simbolo di punteggiatura.

Confrontare le stringhe con gli operatori = e <>

Due stringhe sono uguali solo se sono assolutamente identiche. Come si potrà vedere dall'esempio successivo, l'operatore = calcola sempre un valore False, a meno che si confrontino due stringhe assolutamente identiche, come `"a" = "a"`.

Operazione	Valore dell'operazione
"a" = "a"	True
"a" = "A"	False
"a" = "aa"	False

Nell'esempio seguente, tuttavia, potrete vedere come l'operatore diverso da <> calcoli sempre un valore True, a meno che confronti due stringhe assolutamente identiche, come `"Arca" <> "Arca"`.

Operazione	Valore dell'operazione
"A" <> "a"	True
"Arca" <> "Arca"	False

Durante la comparazione delle stringhe, Visual Basic considera sempre i caratteri maiuscoli e minuscoli come entità completamente diverse.

Confrontare due stringhe con gli operatori >, >=, < e <=

Quando si confrontano le stringhe, Visual Basic calcola il codice di carattere ANSI di ciascuna lettera di ogni stringa, a cominciare dalla prima. La stringa con i caratteri caratterizzati da un codice ANSI superiore viene considerata maggiore.

Per esempio, la lettera "A" ha un codice di carattere ANSI pari a 65, mentre la lettera "a" ha un codice di carattere ANSI pari a 97. Quindi, osservate la seguente riga:

```
Flag = ("Aereo" < "aardvark")
```

Dato che la prima lettera di `"Aereo"` ha un codice di carattere minore rispetto alla prima lettera di "aardvark", Visual Basic considera il valore di `"Aereo"` inferiore al valore di `"aardvark"`, per cui il valore di `Flag` sarebbe True.

Ora, prendete in considerazione il seguente esempio:

```
Flag = ("aereo" < "aardvark")
```

In questo caso, il valore di `Flag` è False. In che modo Visual Basic decide che `"aereo"` è minore di `"aardvark"`? Innanzitutto, Visual Basic calcola il codice di carattere ANSI della prima lettera di ogni stringa e dato che entrambe cominciano con la "a", Visual Basic cerca la seconda lettera. Dato che la "e" ha un codice di carattere ANSI superiore alla "a", `"aereo"` viene considerato maggiore di `"aardvark"` e di conseguenza `Flag` risulta False.

Mettete alla prova le vostre nuove conoscenze

1. Qual è la differenza tra l'operatore / e l'operatore \?

a. Uno viene chiamato barra obliqua e l'altro barra obliqua inversa. Per il resto, assomigliano entrambi a errori di stampa.

b. L'operatore / divide due numeri e l'operatore \ li rimetti insieme.

c. L'operatore / calcola un numero a virgola mobile (decimale), come 3,54, mentre l'operatore \ calcola un numero intero, come 5 o 34.

d. L'operatore / non funziona per cui dovete utilizzare al suo posto l'operatore \.

2. La seguente istruzione è True o False?

```
"aeroplano" < "areoplano"
```

a. False, perché non so cosa pensare, ma questa potrebbe essere la risposta giusta.

b. True, perché la seconda lettera di aeroplano ha un valore minore rispetto alla seconda lettera di areoplano.

c. True e False, perché sto scommettendo pro e contro.

d. False, perché aeroplano è un modo antiquato di dire areoplano, per cui le due stringhe sono assolutamente identiche.

Infine, consideriamo quest'ultimo esempio:

```
Flag ("aereo" < "aereoplano")
```

In questo esempio, il valore di `Flag` è True. Le prime cinque lettere di ciascuna stringa sono identiche ma la sesta lettera no. Dato che `"aereo"` non possiede una sesta lettera, mentre `"aereoplano"` sì, `"aereoplano"` viene considerato maggiore di `Flag` e risulta True.

Precedenza

Con tutti questi operatori che affollano i comandi BASIC, vediamo che cosa succede se vengono ammassati così:

```
Caos = 4 / 7 + 9 * 2
```

Se siete riusciti a indovinare che il valore di `Caos` è 18,57143, congratulazioni! Come fa Visual Basic a gestire una tale istruzione? Innanzitutto, Visual Basic calcola gli operatori con una priorità superiore o *precedenza*.

Non tutti gli operatori sono uguali: alcuni godono del diritto di precedenza sugli altri, ovvero vengono considerati prima. Nella Tabella 17.4 viene illustrato l'ordine

in cui Visual Basic considera i diversi operatori; più un operatore appare in alto, maggiore è la sua precedenza. Quindi, l'operatore di uguaglianza (=) ha una precedenza superiore rispetto all'operatore minore di (<).

Tabella 17.4 Precedenza degli operatori

Operatore	*Tipo di operatore*
Esponenziale (^)	Aritmetico
Negazione (-)	Aritmetico
Moltiplicazione e Divisione (* e /)	Aritmetico
Divisione intera (\)	Aritmetico
Modulo (Mod)	Aritmetico
Addizione e sottrazione (+ e -)	Aritmetico
Concatenazione di stringhe (&)	Aritmetico
Uguaglianza (=)	Confronto
Diseguaglianza (<>)	Confronto
Minore di (<)	Confronto
Maggiore di (>)	Confronto
Minore o uguale a (<=)	Confronto
Maggiore di o uguale a (>=)	
Like	Confronto
Is	Confronto
Not	Logico
And	Logico
Or	Logico
Xor	Logico
Eqv	Logico
Imp	Logico

Torniamo all'esempio illustrato in precedenza. In che modo Visual Basic calcola il valore di Caos in questa equazione?

```
Caos = 4 / 7 + 9 * 2
```

In che modo Visual Basic calcola un risultato?

1. Dato che la moltiplicazione e la divisione hanno la precedenza rispetto all'addizione, Visual Basic cerca dapprima gli operatori di moltiplicazione e di divisione.
2. Dato che la moltiplicazione e la divisione hanno la stessa precedenza, Visual Basic comincia con il primo da sinistra. Di conseguenza, Visual Basic calcola il valore di `4 / 7` e ottiene 0,57143. A questo punto, l'equazione si è ridotta a:

   ```
   Caos = 0.575143 + 9 * 2
   ```

3. Visual Basic vede che l'operatore di moltiplicazione ha una precedenza superiore rispetto all'operatore di addizione e quindi calcola il valore di `9 * 2` e ottiene 18. Ora l'equazione è:

   ```
   Caos = 0.57143 + 18
   ```

Il valore finale di `Caos` è 18.57143

Ma che cosa succede se in realtà volevate che Visual Basic eseguisse la somma prima di calcolare la divisione e la moltiplicazione? Per chiarezza e per essere certi di ottenere il risultato desiderato, è meglio racchiudere sempre tra parentesi le operazioni specifiche dell'equazione, come illustrato nel seguente esempio:

```
Caos = 4 / (7 + 9) * 2
```

Ecco come Visual Basic calcola il risultato.

1. Le parentesi dicono a Visual Basic di sommare innanzitutto `7 + 9`; la somma genera la seguente equazione:

   ```
   Caos = 4 / 16 * 2
   ```

2. Dato che gli operatori di divisione e di moltiplicazione hanno la stessa precedenza, Visual Basic comincia con l'operatore più a sinistra, quindi calcola 4 / 16 e ottiene il risultato 0,25. Ora l'equazione è:

   ```
   Caos = 0.25 * 2
   ```

3. Infine Visual Basic esegue la moltiplicazione e attribuisce il valore 0,5 alla variabile `Caos`.

Ogni volta che si utilizzano due o più operatori, è necessario servirsi delle parentesi per maggiore chiarezza e per assicurarsi che Visual Basic calcoli tutto nell'ordine desiderato.

Per aiutarvi a capire meglio la precedenza, il CD-ROM comprende un programma che illustra il funzionamento della precedenza con e senza parentesi durante il cacolo di un risultato numerico.

Capitolo 18

Stringhe e altre cose

In questo capitolo

- Convertire i caratteri maiuscoli e minuscoli delle stringhe
- Utilizzare parti di stringhe per ricercare e sostituire le stringhe
- Convertire le stringhe in numeri e i numeri in stringhe
- Convertire le stringhe in valori ASCII

Oltre a manipolare i numeri, il programma dovrà anche manipolare le stringhe. Le *stringhe* sono rappresentate da qualsiasi combinazione di lettere, numeri o simboli che il vostro programma dovrà considerare in modo letterale.

Per esempio, i computer interpretano pedissequamente i numeri telefonici e i numeri delle tessere della mutua, considerandoli come espressioni matematiche. Un computer tipico considererebbe il numero di telefono 02-345276 come l'espressione "Sottrai il numero 345276 dal numero 02".

Per dire al programma di considerare le stringhe in modo letterale, è necessario racchiuderle sempre tra apici, "così". Se volete assegnare la stringa 02-345276 a una variabile, utilizzate gli apici, come nel seguente esempio:

```
Private Sub Count()
  Tel = "02-345276"
End Sub
```

Se si dimentica di aggiungere gli apici, Visual Basic tenterà di interpretare la stringa di numeri come un comando vero e proprio. In questo caso Visual Basic tenderebbe a sottrarre 345276 da 02.

Dopo che saranno stati specificati dei dati particolari come una stringa, Visual Basic offrirà un'infinità di sistemi per esaminare, manipolare e accorciare questa stringa.

Manipolare le stringhe

Non si è costretti a utilizzare le stringhe così come appaiono in origine. Si possono convertire i caratteri maiuscoli e minuscoli di una stringa, utilizzare elementi di una stringa per ricercare e sostituire altre stringhe e accorciare una stringa eliminando gli spazi superflui.

Determinare la lunghezza di una stringa

Una stringa è lunga quanto il numero di caratteri, spazi compresi, in essa contenuti. Per determinare la lunghezza di una stringa, utilizzate il seguente comando BASIC:

```
NomeVariabile = Len("Stringa")
```

Per esempio:

```
Private Sub Command1_Click()
  Dim Nome As String
  Dim LunghezzaNome As Integer
  Nome = "Olimpia la gatta"
  LunghezzaNome = Len(Nome)
End Sub
```

In questo caso, la lunghezza della stringa `"Olimpia la gatta"` è 16 (14 lettere e 2 spazi); tale valore viene quindi attribuito alla variabile denominata `LunghezzaNome`.

Convertire i caratteri da maiuscoli a minuscoli

Per convertire tutta una stringa in caratteri minuscoli, utilizzate il seguente comando BASIC:

```
LCase("Stringa")
```

Per esempio:

```
Private Sub Form_Click()
  Dim Nome As String, Minuscolo As String
  Nome = "TUTTO QUESTO NON HA FORSE UN ASPETTO SGRADEVOLE?"
  Minuscolo = LCase(Nome)
End Sub
```

In questo caso il valore di `Minuscolo` è rappresentato dalla seguente stringa:

```
tutto questo non ha forse un aspetto sgradevole?
```

Si noti che il comando `LCase` agisce soltanto sulle lettere (credevate di poter convertire in caratteri minuscoli un punto di domanda?). Per convertire tutta una stringa in caratteri maiuscoli, utilizzate il seguente comando BASIC:

```
UCase("Stringa")
```

Per esempio:

```
Private Sub Form_Click()
  Dim Nome As String, Maiuscolo As String
  Nome = "parlate sottovoce"
  Maiuscolo = UCase(Nome)
End Sub
```

In questo caso il valore di `Maiuscolo` è rappresentato dalla seguente stringa:

```
PARLATE SOTTOVOCE
```

Invertire le stringhe

Visual Basic 6 comprende un comando nuovo che consente di invertire le stringhe, utile per giocare con i *palindromi* (vale a dire quelle frasi che leggono in maniera identica sia da destra che da sinistra). Per invertire una stringa, utilizzate il seguente comando BASIC:

```
StrReverse("Stringa")
```

Per esempio:

```
Private Sub Form_Click()
Dim Frase As String
  Frase = "Anna Panna"
  Frase = StrReverse(Frase)
End Sub
```

In questo caso, il valore di Frase è la stringa seguente:

```
annaP annA
```

Estrarre i caratteri da una stringa

Talvolta una stringa contiene più informazioni di quelle desiderate. Per esempio, potreste avere memorizzato il nome completo di una persona nella variabile `NomeCompleto`, come viene illustrato nel seguente esempio:

```
NomeCompleto = "Gianni Bronco"
```

Per estrarre i caratteri a partire dalla sinistra della stringa, utilizzate il seguente comando BASIC:

```
Left(NomeCompleto, N)
```

Il comando precedente dice: "Vedi quella stringa chiamata `NomeCompleto`? Considera solo un numero N di caratteri a partire dalla sinistra". Per esempio:

```
Private Sub Form_Click()
  Dim NomeCompleto As String, Nome As String
  NomeCompleto = "Gianni Bronco"
  Nome = Left(NomeCompleto, 6)
End Sub
```

Nell'esempio precedente, il valore di `Nome` è `Gianni`.

Per estrarre i caratteri a partire da destra della stringa, utilizzate il seguente comando BASIC:

```
Right(NomeCompleto, N)
```

Questo comando dice: "Vedi quella stringa chiamata `NomeCompleto`? Considera solo un numero N di caratteri a partire da destra". Per esempio:

```
Private Sub Form_Click()
  Dim NomeCompleto As String, Cognome As String
  NomeCompleto = "Gianni Bronco"
  Cognome = Right(NomeCompleto, 6)
End Sub
```

In questo esempio, il valore di `Cognome` è `Bronco`.

Questo comando dice: "Vedi quella stringa chiamata NomeCompleto? Conta X caratteri da sinistra ed elimina i successivi Y caratteri. Per esempio;

```
Private Sub Form_Click()
Dim NomeCompleto As String, Mezzo As String
  NomeCompleto = "Gianni P. Bronco"
  Mezzo = Mid(NomeCompleto, 8, 2)
     End Sub
```

In questo esempio, il valore di Mezzo è P. (compreso il punto).

Trovare una parte di stringa utilizzando un'altra stringa

Se una stringa è nascosta nel mezzo di un'altra stringa, è possibile individuarne la posizione utilizzando il seguente comando BASIC:

```
InStr("StringaDestinazione" , "StringaDesiderata")
```

Questo comando restituisce un numero che definisce la posizione esatta da sinistra in cui comincia la "`StringaDesiderata`" all'interno della "`StringaDestinazione`". Per esempio:

```
Private Sub Form_Click()
  Dim NomeCompleto As String
  NomeCompleto = "Gianni Nino Bronco"
  Posizione = InStr(NomeCompleto, "Nino")
End Sub
```

In questo caso il valore di `Posizione` è 8.

Se la stringa che volete non fosse presente all'interno della stringa sottoposta a ricerca, il comando `InStr` restituirebbe 0.

Quando si ricerca una stringa all'interno di un'altra stringa, è necessario utilizzare i caratteri maiuscoli e minuscoli esatti. Per esempio, il seguente comando restituirebbe 0:

```
InStr("Gianni Nino Bronco", "NINO")
```

In questo caso, `"Nino"` non è uguale alla stringa `"NINO"` per cui `InStr` restituisce 0. In pratica, uno 0 è il modo utilizzato da Visual Basic per dire: "Mi spiace, ma non riesco a trovare questa stringa esatta da nessuna parte".

Sostituire parte di una stringa con un'altra stringa

Nel caso si senta il bisogno creativo di scrivere un programma di elaborazione testi utilizzando Visual Basic (completo delle funzioni di ricerca e sostituzione), utilizzate il seguente comando BASIC:

```
Mid("StringaDestinazione", Posizione) = "NuovaStringa"
```

Questo comando dice: "Vedi quella stringa denominata `StringaDestinazione`? Spostati sul punto definito da `Posizione`, partendo da sinistra, e inserisci la stringa denominata `NuovaStringa`".

Naturalmente, quando inserite una nuova stringa in una stringa esistente, dovete fare attenzione. Per esempio, consideriamo il seguente codice:

```
NomeCompleto = "Gianni Nino Bronco"
Mid(NomeCompleto, 8) = "Vaniglia"
```

Ecco come Visual Basic interpreta questo codice.

Innanzitutto, Visual Basic attribuisce la stringa `"Gianni Nino Bronco"` alla variabile denominata `NomeCompleto`.

Quindi Visual Basic cerca la stringa `"Gianni Nino Bronco"`, trova l'ottavo carattere a partire da sinistra e inserisce la stringa `"Vaniglia"`, sostituendo la stringa originale che inizia con l'ottavo carattere. Ecco che cosa succede.

Gianni Nino Bronco	(stringa originale)
^	(ottavo carattere a partire da sinistra)
Gianni Vaniglianco	(nuova stringa)

Dopo che sarà stato detto a Visual Basic di sostituire parte di una stringa con un'altra, Visual Basic avrà un eccesso di zelo ed eliminerà tutto ciò che potrebbe ostacolare la nuova stringa.

Togliere gli spazi dalle stringhe

Le stringhe non sono sempre belle e ordinate. Talvolta ci sono spazi inutili all'inizio o alla fine, come viene illustrato nei seguenti esempi:

```
"      Questo è un esempio di spazi all'inizio"
"Questo è un esempio di spazi alla fine       "
```

Per esempio:

```
Private Sub Form_Click()
  Dim NomeCompleto As String
  NomeCompleto = "      Gianni Bronco"
  NomeCompleto = LTrim(NomeCompleto)
End Sub
```

Senza gli spazi iniziali, il valore di `NomeCompleto` sarebbe `"Gianni Bronco"`.

Per eliminare gli spazi alla fine, utilizzate il seguente codice BASIC:

```
RTrim("StringaDestinazione")
```

Mettete alla prova le vostre nuove conoscenze

1. In che modo potete dire al vostro programma di trattare le stringhe in modo letterale?

a. È sufficiente dire: “Non sto scherzando, davvero!”

b. Aggiungendo la parola TrattaLetteralmente alla riga di codice in cui risiede la stringa.

c. Racchiudendo la stringa tra apici.

d. Parlando in maniera chiara e scandendo bene le parole, in modo da evitare incomprensioni.

2. Che cosa fa la seguente riga di codice?

```
Trovato = InStr(StringaDestinazione", "StringaDesiderata")
```

a. Questa riga mostra un elenco dei dieci criminali più ricercati, elenco peraltro affisso anche all’ufficio postale.

b. Questa riga di codice inserisce un’icona a “occhio di bue”, che mostra dove mirare con il fucile quando vi sentite frustrati.

c. Questa riga restituisce un numero, memorizzato nella variabile `Trovato`, che definisce la posizione esatta, da sinistra, nella quale è possibile trovare `"StringaDesiderata"` all’interno di `"StringaDestinazione"`.

d. Tutto quanto è stato detto qui sopra.

Per esempio:

```
Private Sub Form_Click()
  Dim NomeCompleto As String
  NomeCompleto = "Gianni
  NomeCompleto = RTrim(NomeCompleto)
  NomeCompleto = NomeCompleto & " " & "Bronco"
End Sub
```

Nell’esempio precedente, il comando `RTrim` elimina gli spazi finali in modo che il valore di `NomeCompleto` sia semplicemente `"Gianni"`. Quindi l’ultimo comando somma il valore di `NomeCompleto` (`"Gianni"`), uno spazio vuoto (`" "`) e la stringa `"Bronco"` per creare la stringa `"Gianni Bronco"`. Nel caso si abbiano spazi sia all’inizio che alla fine, è possibile unire i due comandi nel seguente modo:

```
LTrim(RTrim("StringaDestinazione"))
```

Questo comando dice: “Innanzitutto togli tutti gli spazi finali e poi togli tutti quelli iniziali”. Un metodo ancora più semplice consiste nell’utilizzare il seguente comando BASIC:

```
Trim("StringaDestinazione")
```

Per esempio:

```
Private Sub Form_Click()
  Dim NomeCompleto As String
  NomeCompleto = "    Gianni Nino    "
  NomeCompleto = Trim(NomeCompleto)
  NomeCompleto = NomeCompleto & " " & "Bronco"
End Sub
```

Il comando `Trim` elimina sia gli spazi iniziali che quelli finali in un colpo solo, quindi l'ultimo comando unisce la stringa `"Gianni Nino"`, una stringa vuota (`" "`) e la stringa `"Bronco"` per creare la stringa `"Gianni Nino Bronco"`.

Convertire stringhe e valori

Visual Basic gestisce numeri e stringhe in modo diverso. Talvolta potrebbe essere necessario convertire una stringa in un numero, in modo da poterla utilizzare per dei calcoli; oppure convertire un numero in una stringa, in modo da poterla modificare; o, ancora, convertire una stringa nel suo valore ASCII o ANSI.

Convertire una stringa in un numero

Che cosa succede se esiste una casella di testo nella quale gli utenti possono digitare le loro paghe orarie? Purtroppo, la proprietà `Text` di qualsiasi casella di testo memorizza i dati come stringa e non come numero; per convertire questa stringa in un numero, va utilizzato uno dei seguenti comandi BASIC:

```
CDbl("StringaDestinazione")
CSng("StringaDestinazione")
```

Il primo comando dice: "Prendi la stringa denominata `StringaDestinazione` e convertila in un numero di tipo Double".

Il secondo comando dice: "Prendi la stringa denominata `StringaDestinazione` e convertila in dato di tipo Single".

Per esempio:

```
Private Sub Form_Click()
  Dim Numero As Double
  Numero = CDbl (txtPagaOraria.Text)
End Sub
```

Ecco in che modo Visual Basic interpreta il codice.

1. La prima istruzione dice: "Dichiara una variabile denominata `Numero` come dato di tipo `Double`.
2. La seconda istruzione dice: "Recupera la stringa memorizzata nella proprietà `Text` di una casella di testo denominata `PagaOraria` e convertila in un numero di tipo Double".
3. Infine, il valore memorizzato nella proprietà `txtPagaOraria.Text` viene attribuito alla variabile `Numero`".

Se l'utente ha digitato **50000** nella casella di testo `txtPagaOraria`, il valore di `Numero` sarà `50000`.

Se invece nella casella di testo `txtPagaOraria` l'utente ha digitato **50000 all'ora** oppure **La mia paga oraria è 50000**, Visual Basic comincerà a brontolare e dichiarerà un errore, dato che `CDbl` e `CSng` non sono in grado di gestire i caratteri.

Convertire un numero in una stringa

Che cosa succede se si ha un numero e si deve convertire in una stringa in modo da poterlo utilizzare come tale? Va utilizzato il seguente comando BASIC:

```
CStr(Numero)
```

Questo comando dice: "Prendi il numero rappresentato da Numero e trasformalo in una stringa".

Per esempio, Visual Basic considera queste due indicazioni come creature del tutto diverse:

```
10       ' Questo è un numero
"10"     ' Questa è una stringa
```

I comandi di seguito riportati convertono un numero in una stringa:

```
CStr(10)        ' La stringa è " 10"
CStr(10,5)      ' La stringa è " 10.5"
CStr(-10)       ' La stringa è "-10"
```

Quando Visual Basic converte un numero in una stringa, alla stringa viene aggiunto uno spazio vuoto iniziale se si tratta di un numero positivo oppure il segno meno (-) se si tratta di un numero negativo.

Convertire una stringa nel corrispondente valore ASCII

In qualità di programmatori, prima o poi dovrete sapere a memoria la tabella dei codici ASCII, quindi sarebbe meglio trovarne una copia e appenderla di fianco al vostro computer.

Una tabella ASCII mostra il codice che il computer utilizza per rappresentare la maggior parte dei caratteri di cui si ha bisogno. Per esempio, la lettera *A* ha il valore ASCII di 65, mentre la lettera *a* ha un valore ASCII di 97.

Ogni volta che avrete bisogno del valore ASCII di una stringa composta da un unico carattere, potete utilizzare il seguente comando BASIC:

```
Asc("Carattere")
```

Di seguito potete vedere come convertire un carattere nel suo valore ASCII:

```
X = Asc ("A") ' X = 65
X = Asc ("a") ' X = 97
```

Convertire un valore ANSI in una stringa

Microsoft Windows non utilizza la tabella ASCII ma la tabella ANSI, che è praticamente identica alla tabella ASCII (nel Capitolo 17 è stata illustrata la tabella ANSI).

Ecco il comando BASIC per utilizzare un valore ANSI:

```
Chr("Carattere")
```

Le uniche volte in cui avrete bisogno di utilizzare un valore ANSI saranno i casi in cui vi serviranno codici di controllo speciali, come gli avanzamenti riga, i ritorni a capo e le nuove righe.

I seguenti comandi mostrano alcuni valori ANSI utilizzati di frequente:

```
AvanzamentoRiga = Chr(10)
AvanzamentoPagina = Chr(12)
RitornoCarrello = Chr(13)
```

Utilizzando questi splendidi comandi, potete assicurarvi che le vostre stringhe abbiano l'aspetto desiderato prima di visualizzarle in una casella di testo o in un'etichetta. Oppure potete semplicemente divertirvi con parole e numeri e fingere di lavorare seriamente.

Per capire in che modo Visual Basic manipola le stringhe, esaminate il programma ELIZA.VBP presente sul CD-ROM. Questo programma è simile al famoso programma di intelligenza artificiale ELIZA che mima uno psichiatra che non fa che ripetere parte di ciò che l'utente digita.

Capitolo 19

Definire le costanti e utilizzare i commenti

In questo capitolo

- Denominare e calcolare le costanti
- Dichiarare la visibilità delle costanti
- Creare e utilizzare tre tipi di commenti

Una *costante* è un valore fisso che non muta mai, a prescindere da che cosa succede al programma. I numeri, le stringhe e le date possono essere costanti.

Ma perché utilizzare le costanti? Esistono numerose buone ragioni, una di queste è dare un senso al vostro programma prima di iniziare a scriverlo.

Per esempio, supponete di voler scrivere un programma per pagare i dipendenti in base allo stipendio minimo attuale. Se la paga minima fosse 15.000, dovreste utilizzare il numero 15.000 in tutto il vostro programma.

Purtroppo, il numero 15.000 non significa nulla di per sé, è solo un valore. E peggio ancora, se la paga minima variasse da 15.000 a 16.500, dovrete modificare 15.000 in 16.500 in tutto il programma.

Per risolvere questi problemi, si possono utilizzare le costanti; costante è semplicemente una parola che rappresenta un valore specifico. Una costante non solo descrive il significato del valore in italiano ma permette di modificare il valore stesso rapidamente e facilmente.

Attribuire un nome alle costanti

I nomi delle costanti devono soddisfare i seguenti criteri.

- ✓ Cominciare con una lettera.

- ✓ Essere composti al massimo da 40 caratteri.
- ✓ Contenere solo lettere, numeri e caratteri di sottolineatura (_); gli spazi e i simboli di punteggiatura non sono ammessi.
- ✓ Essere una parola qualsiasi, ma diversa dalle parole chiave riservate di Visual Basic.

Per fare in modo che il nome di una costante salti all'occhio, utilizzate tutte lettere maiuscole. Per esempio, i seguenti nomi sono accettabili:

```
ETÀ
MIO_COMPLEANNO
PAGA_MINIMA
CAPACITÀ_SCIALUPPA_SALVATAGGIO
```

Per fornire ulteriori informazioni sul tipo di dati rappresentati da una costante, aggiungete al nome della costante un prefisso costituito da tre lettere, così:

```
intETA'     (int rappresenta un valore di tipo Integer)
curPAGA_MINIMA (cur rappresenta un valore di tipo Currency)
sngGPA      (sng rappresenta un valore di tipo Single)
```

La Tabella 19.1 elenca i prefissi di tre lettere suggeriti da Microsoft per le costanti (o anche per le variabili).

Tabella 19.1 Prefissi di tre lettere da utilizzare nei nomi delle costanti e nelle variabili

Tipo di dati	*Prefisso*	*Esempio*
Boolean	bln	blnEGiàMorto
Byte	byt	bytQuestoNumero
Currency	cur	curAumento del Capo
Date	dtm	dtmAnniversario
Double	dbl	dblAltezza
Integer	int	intNumeroTotale
Long	lng	lngLarghezza
Single	sng	sngMedia
String	str	strMioNome
Variant	vnt	vntQualsiasiCosa

Dichiarare le costanti

Prima di poter utilizzare una costante, questa va *dichiarata*. Per fare ciò, è sufficiente attribuirle un nome e assegnarle un valore specifico, come uno dei seguenti:

- ✓ Numeri
- ✓ Stringhe
- ✓ Date

Il codice di seguito riportato dichiara costanti costituite da numeri, stringhe e date:

```
Private Sub Command1_Click()
  Const intETA = 21
  Const strAZIENDA = "Manifatture S.p.A."
  Const dtmNATALE = #25 December 1995#
End Sub
```

Tutte le dichiarazioni di costanti devono essere poste all'inizio delle procedure di gestione degli eventi. Invece di digitare una dichiarazione di costante su ogni riga, si possono raggruppare e separare mediante virgole, come illustrato nel seguente codice:

```
Private Sub Command1_Click()
  Const intETA = 21, Const strAZIENDA = "Manifatture S.p.A."
End Sub
```

Si noti che le *costanti con valore numerico* sono costituite solo da numeri, le *costanti con stringhe* sono rappresentate da qualsiasi cosa racchiusa tra apici ("") mentre le *costanti con date* sono racchiuse dal simbolo #.

Ecco alcuni sistemi per visualizzare le date:

```
#12-25-95#
#December 25, 1995#
#Dec-25-95#
#25 December 1995#
```

Calcolare una costante

Le costanti rappresentano di solito un valore fisso ma possono essere anche valori matematici fondati su altre costanti. Per esempio:

```
Const intETA_PENSIONE = 65
Const sngMETA_STRADA = ETA_PENSIONE / 2
```

In questo caso il valore della costante intETA_PENSIONE è 65, mentre il valore della costante sngMETA_STRADA è 65/2, ovvero 32,5.

Utilizzare le costanti

Dopo avere dichiarato una costante, è possibile utilizzarla come qualsiasi altro valore. Si osservi il seguente esempio:

```
Const curPAGA_MINIMA = 15000
Stipendio = curPAGA_MINIMA * 20
```

Ecco come Visual Basic interpreta questo codice.

1. Il primo comando dice: "Crea una costante denominata curPAGA_MINIMA e imposta il suo valore a 15000".
2. Il secondo comando dice: "Moltiplica sempre il valore di curPAGA_MINIMA per 20 e memorizza questo valore nella variabile denominata Stipendio". In questo caso, il valore di curPAGA_MINIMA è 15000, quindi moltiplicando 15000 per 20, si ottiene 300000. Visual Basic memorizza questo valore in Stipendio.

Visibilità delle costanti

Visual Basic permette di dichiarare la visibilità delle costanti utilizzando i seguenti tre metodi.

- ✓ Locale.
- ✓ Modulo.
- ✓ Pubblica.

Costanti locali

Una costante *locale* può essere utilizzata solo all'interno della procedura nella quale è stata dichiarata. Lo scopo delle costanti locali è isolare delle costanti specifiche nella procedura in cui vengono utilizzate.

Si dichiari una costante locale all'interno di una procedura di gestione degli eventi nel seguente modo:

```
Private Sub Command1_Click()
  Const intLIMITE_VELOCITA = 55
End Sub
```

Una costante locale può essere utilizzata solo nella procedura di gestione degli eventi nella quale è stata dichiarata. Ma che cosa si deve fare quando si vuole creare una costante condivisibile da una o più procedure di gestione degli eventi? In questo caso va creata una costante di modulo.

Costanti di modulo

Una *costante di modulo* può essere utilizzata soltanto da una procedura di gestione degli eventi memorizzata nello stesso file.

Per dichiarare una costante di modulo, utilizzate la procedura di seguito riportata.

1. **Aprite la finestra Codice premendo F7, selezionando Visualizza - Codice oppure facendo doppio clic su un punto qualsiasi della form.**
2. **Fate clic sulla casella di riepilogo Oggetto nella finestra Codice e selezionate (generale).**
3. **Fate clic sulla casella di riepilogo Procedura e selezionate (dichiarazioni).**
4. **Ora digitate la dichiarazione della costante utilizzando l'istruzione `Const` nel seguente modo:**

```
Const intETA-ALCOLICI = 21
```

Le costanti di modulo sono utili per condividere il valore di una costante tra una o più procedure di gestione degli eventi, tuttavia la costante resta limitata alle procedure di gestione degli eventi memorizzate nello stesso file; se si vuole una costante che sia utilizzabile da qualsiasi procedura del programma, bisogna creare una costante pubblica.

Per aiutarvi a identificare le costanti modulo sepolte nel codice, Microsoft raccomanda di porre la lettera m davanti al nome della costante, così:

```
Const mintETA-ALCOLICI = 21
```

Costanti pubbliche

Una costante *pubblica* (definita anche costante globale) può essere comodissima da utilizzare, dato che qualsiasi procedura del programma Visual Basic potrà accedervi. Tuttavia i programmatori in gamba utilizzano le costanti pubbliche solo quando è assolutamente necessario: è considerata una pratica errata riempire il programma di costanti pubbliche che verranno utilizzate solo da poche procedure.

L'uso di costanti pubbliche costituisce un esempio di programmazione maldestra poiché la modifica di una costante pubblica può ripercuotersi sull'intero programma. I programmatori esperti arrossiranno imbarazzati se vi sorprende-

ranno a utilizzare inutilmente costanti pubbliche e con ogni probabilità non sarete più invitati a nessuna festa per programmatori seri.

Le costanti pubbliche devono essere dichiarate nel file (modulo) BAS. Per dichiarare una costante pubblica, si utilizzi la procedura di seguito riportata.

1. **Aprite la finestra Progetto.**

   A questo scopo, premete CTRL+R sul file (modulo) BAS in cui volete inserire la costante pubblica. Se dovete creare un file BAS, fate clic sull'icona Modulo oppure selezionate Inserisci modulo dal menu Progetto.

2. **Premete F7, selezionate Visualizza-Codice oppure fate doppio clic su un punto qualsiasi della form per aprire la finestra Codice.**
3. **Fate clic sulla casella di riepilogo Oggetto nella finestra Codice e selezionate (generale).**
4. **Fate clic sulla casella di riepilogo Procedura e selezionate (dichiarazioni).**
5. **Digitate la costante pubblica utilizzando il comando `Public`, come illustrato nell'esempio seguente:**

```
Public Const LIMITE_VELOCITA = 55
```

Per aiutarvi a identificare le costanti pubbliche (o globali) sepolte nel codice, Microsoft raccomanda di porre la lettera g davanti al nome della costante, così:

```
Const gsngLIMITE_VELOCITA = 55
```

Se si vuole vedere un elenco di tutte le costanti pubbliche che Visual Basic ha già definito, si utilizzi la procedura di seguito riportata.

1. **Selezionate ?-INDE*CE.**

   Appare una finestra di dialogo.

2. **Fate clic sulla scheda Indice.**
3. **Digitate** costanti, **nella casella di testo Type in the keyword to find.**
4. **Nella categoria Costanti, fate clic su Visul Basic.**
5. **Fate clic su Visualizza.**

   Visual Basic visualizza la finestra di dialogo Costanti di Visual Basic.

Utilizzare i commenti

Quando state codificando il vostro programma (notate l'impiego astuto della parola *codificando*), potreste anche avere le idee chiare sul suo funzionamento. Pur-

troppo, se mettete il programma da parte e lo modificate cinque anni dopo, potreste avere dimenticato perché avete scritto determinati comandi e addirittura in che modo alcuni di quei comandi funzionano.

Per questo motivo aggiungete numerosi commenti ai programmi che si creano. I *commenti* sono brevi descrizioni che i programmatori aggiungono ai loro programmi per spiegare il significato di determinati comandi o per indicare che cosa dovrebbe succedere in una parte particolare del programma.

Per quanto riguarda il computer, i commenti non fanno assolutamente nulla per aiutare o ostacolare il funzionamento del programma, ma dal punto di vista del programmatore i commenti contribuiscono a spiegare come e perché funziona un programma.

Creare dei commenti

Visual Basic permette di creare dei commenti utilizzando il simbolo dell'apostrofo ('), seguito da qualsiasi cosa digitiate. Di seguito viene illustrato un commento valido:

```
Private Sub Command1_Click()
   ' Questa procedura di gestione degli eventi non fa
assolutamente nulla
End Sub
```

Per quanto riguarda Visual Basic, tutto ciò che appare a destra dell'apostrofo viene ignorato dal computer.

I commenti possono apparire su righe separate oppure possono far parte di un'altra riga, come viene illustrato nel seguente esempio:

```
Private Sub Command1_Click()
  X = Y * 2 ' Y rappresenta il numero di impiegati
End Sub
```

Si possono anche raggruppare più commenti utilizzando diverse righe:

```
Private Sub Command1_Click()
  Y = 200 ' Y rappresenta il numero di impiegati
  X = Y * 2
  ' X rappresenta il numero di impiegati che vorrebbero
tagliare le gomme dell'auto del capo
End Sub
```

È sufficiente ricordare che tutto ciò che compare a destra dell'apostrofo viene ignorato dal computer e viene considerato un commento.

Inserire commenti per aumentare la leggibilità

La ragione principale per utilizzare i commenti è rendere più comprensibili i programmi ed è per questo che la maggior parte dei programmatori inserisce dei commenti all'inizio di ogni procedura.

Questi commenti spiegano quali dati recupera la procedura e che calcoli vengono eseguiti. Osservando i commenti, chiunque può capire velocemente che cosa fa la procedura senza dover decifrare diverse righe di difficile codice BASIC. Per esempio, riuscite a capire che cosa fa la seguente procedura di gestione degli eventi?

```
Private Sub Command1_Click()
  A = SQR(B ^ 2 + C ^ 2)
End Sub
```

Per rendere la funzione di questa procedura molto più semplice da capire, è sufficiente aggiungere una serie di commenti:

```
Private Sub Command1_Click()
  'La seguente equazione utilizza il teorema di Pitagora per
calcolare la lunghezza del lato di un triangolo rettangolo se
è nota la lunghezza dei due cateti. In questo caso, la
lunghezza di un cateto del triangolo è rappresentata da B
mentre la lunghezza del secondo cateto del triangolo è
rappresentata da C.
  A = SQR((B ^ 2 + C ^ 2))
End Sub
```

Se diverse persone partecipano alla scrittura delle procedure, si possono utilizzare i commenti per segnalare il nome del programmatore e la data in cui ciascuna procedura è stata modificata (in questo modo si saprà chi rimproverare quando la procedura non funziona). Per esempio:

```
Private Sub Command1_Click()
  ' Programmatore: Gianni Bronco
  ' Ultima modifica: 1/1/80 (l'orologio del nostro computer
non funziona)
  A = SQR((B ^ 2 + C ^ 2))
End Sub
```

Ovviamente lo svantaggio è che se ci si lascia prendere la mano i commenti possono creare confusione più che aiutare, come i cartelli dell'autostrada. Ricordate: fornite solo le informazioni necessarie, non troppe, in modo da evitare che la gente si addormenti leggendole. Non state scrivendo un romanzo classico ma solo una breve descrizione che gli altri dovranno capire.

Noiosa poltiglia tecnica

1. **Perché dovreste aggiungere dei commenti al vostro programma?**

   a. Per riassumere e spiegare come funziona il codice BASIC.

   b. Per esercitare le vostre capacità letterarie e dimostrare che anche i programmatori sanno scrivere.

   c. Per dimostrare che avete qualcosa da dire oltre ai comandi BASIC.

   d. Per lasciare messaggi in codice che gli altri programmatori dovranno decifrare.

2. **Esprimete un commento sulla semplicità e la brevità di questa lezione.**

   a. Ottimo! Così posso chiudere tutto e tornare a casa presto.

   b. Perché anche le altre lezioni del libro non possono essere altrettanto brevi e semplici?

   c. Non sono ancora in grado di scrivere un programma ma so come utilizzare i commenti. Forse dovrei cercare un impiego come commentatore.

   d. I commenti sono una bella cosa. Se possiamo inserire dei commenti nel nostro programma significa che siamo in grado di scrivere programmi utilizzando dei pacchetti di elaborazione testi?

Commenti per aumentare la chiarezza

Se il programma contiene molto codice BASIC, si possono utilizzare i commenti e le righe vuote per rendere il codice più semplice da leggere. Per esempio, potreste suddividere il codice in spezzoni per evidenziare la funzione di ciascun blocco:

```
Private Sub Command1_Click()
  Const TASSO_INTERESSE = .055 ' tasso di interesse del 5,5%
  Dim Msg As String ' Dichiara Msg come variabile di tipo
stringa
  BilancioBanca = 1000000
  BilancioBanca = BilancioBanca * TASSO_INTERESSE
  ' Sottrazione spese bancarie
  SpeseBanca = BilancioBanca * 2
  BilancioBanca = BilancioBanca - SpeseBanca
  ' Visualizza una casella dei messaggi che informa l'utente
del proprio debito verso la banca
  Msg = "Per favore, pagate il seguente ammontare: " & -
BilancioBanca
  MsgBox Msg, vbCritical, "Ammontare dovuto"
End Sub
```

Come si vede dall'esempio precedente, è possibile inserire dei ritorni a capo manuali per aggiungere righe vuote tra i blocchi di codice, rendendo così più evidente la funzione di ciascun blocco.

Togliendo tutti i commenti e le righe vuote si ottiene questa versione decisamente più brutta:

```
Private Sub Command1_Click()
  Const TASSO_INTERESSE = .055
  Dim Msg As String
  BilancioBanca = 1000000
  BilancioBanca = BilancioBanca * TASSO_INTERESSE
  SpeseBanca = BilancioBanca * 2
  BilancioBanca = BilancioBanca - SpeseBanca
  Msg = "Per favore, pagate il seguente ammontare: " & -
BilancioBanca
  MsgBox Msg, vbCritical, "Ammontare dovuto"
End Sub
```

Notate come questa versione sembri stipata, proprio come l'armadietto del vostro bagno o il garage.

Commenti di disattivazione

Con i commenti non solo potete aggiungere spiegazioni al vostro programma e inserire separazioni visive nel vostro codice, ma potete anche disattivare temporaneamente uno o più comandi BASIC.

Per esempio, nello scrivere un programma ci si potrebbe accorgere che un comando non funziona come dovrebbe. Per provare il funzionamento del programma senza questo comando, si hanno due possibilità.

- ✓ Cancellare il comando.
- ✓ Escluderlo con un commento.

Se si cancella un comando e poi se ne ha bisogno, tutto va nuovamante digitato. Se invece lo si esclude con un commento, sarà sufficiente cancellare il simbolo dell'apostrofo per reinserire il comando.

L'esempio di seguito riportato contiene una serie di cifre abbastanza lunga:

```
Private Sub Command1_Click()
  X = 3,14 * 650 - (909 / 34,56) + 89,323
End Sub
```

Se cancellaste la seconda riga, doverla digitare di nuovo potrebbe risultare piuttosto noioso; potete invece escluderla trasformandola nel seguente commento:

```
Private Sub Command1_Click()
  ' X = 3,14 * 650 - (909 / 34,56) + 89,323
End Sub
```

Ricordate tutto quello che appare a destra del simbolo di apostrofo viene ignorato dal computer. Quindi per il computer la procedura ora è la seguente:

```
Private Sub Command1_Click()
End Sub
```

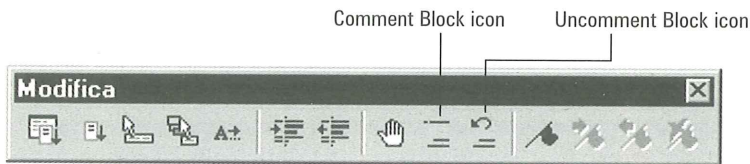
Il fatto di porre l'apostrofo di fronte a questa istruzione, la trasforma in un commento e disattiva il comando BASIC; togliendo il simbolo dell'apostrofo è possibile trasformare rapidamente il commento in un comando BASIC attivo.

Utilizzando i commenti con oculatezza, potete fare in modo che tutti i programmi da voi scritti siano facili da capire sia da voi che dagli altri programmatori. Poi, se davvero intendete sabotare un progetto di programmazione, aggiungete commenti privi di senso oppure tralasciateli del tutto e state a vedere che cosa succede.

Se desiderate commentare rapidamente più righe di codice, procedente nella maniera seguente.

1. **Evidenziate le righe di codice che desiderate trasformare in un commento.**
2. **Selezionate Visualizza-Barre degli strumenti-Modifica.**

   Viene visualizzata la barra degli strumenti Modifica illustrata nella Figura 19.1.

Figura 19.1 La barra degli strumenti Modifica

3. **Fate clic sull'icona Comment Block.**

   Visual Basic aggiunge un apostrofo davanti a ogni riga evidenziata nel Passo 1.

Per togliere dalle righe di commento blocchi di codice, ripetete i passi appena illustrati e, al passo 3, fate clic sull'icona Uncomment Block della barra degli strumenti Modifica.

Capitolo 20

Eliminare gli errori di programmazione

In questo capitolo

- Esaminare i vari tipi di errore
- Caccia grossa
- Eliminare gli errori

Anche se avete già scritto milioni di programmi diversi, molto probabilmente prima o poi commetterete un errore. Potreste scrivere male una parola o dimenticare di digitare un comando. A prescindere dall'attenzione dedicata al programma, quest'ultimo potrebbe non funzionare come dovrebbe; gli errori che impediscono al programma di funzionare a dovere sono detti *bug* (bachi) nello slang informatico.

Tutti i programmi del mondo contengono delle imperfezioni, compresi Netscape Navigator, WordPerfect, Lotus 1-2-3, Quicken e Microsoft Windows 98. L'unica differenza tra gli errori del vostro programma e quelli di un programma commerciale sta nel fatto che nessuno vi paga per eliminare gli errori del vostro programma. Utilizzando criteri di pianificazione, progettazione applicativa e semplice buon senso riuscirete a evitare un buon numero di errori.

Ma non preoccupatevi troppo: molti errori sono abbastanza innocui e, solitamente, non impediscono un utilizzo corretto del programma, sebbene possano rallentare il computer oppure visualizzare di tanto in tanto sullo schermo strani oggetti o colori.

I grossi errori sono molto più devastanti. Per esempio, alla NASA, sembra che un errore abbia fatto esplodere un satellite meteorologico da parecchi milioni di dollari solo perché qualcuno aveva scritto in modo errato un comando all'interno del programma.

Nessuno è perfetto, per cui è impossibile garantire che un programma non contenga errori. Persino un professionista esperto con una laurea in informatica continuerà a scrivere programmi pieni di errori.

Perché nello slang informatico gli errori dei computer vengono definiti bug (bachi)?

Il primo computer al mondo utilizzava dei relè meccanici invece della moderna elettronica. Un giorno il computer smise di funzionare senza alcuna ragione evidente. Gli scienziati controllarono il loro lavoro di programmazione (teoricamente doveva funzionare), il cavo di alimentazione (era ben inserito) e i file all'interno del computer (erano ancora tutti ben collegati). Alla fine, qualcuno notò che una farfalla era rimasta schiacciata tra due relè e impediva a uno di chiudersi completamente. Dato che era stato il cadavere della farfalla a provocare il guasto, i problemi dei computer vennero battezzati *bug* (bachi), una parola più semplice da pronunciare di chihuahua; meno male che nel primo computer non era rimasto incastrato un cane.

Gli errori sono una realtà della vita, proprio come gli scarafaggi in cucina: non è mai possibile sbarazzarsene totalmente, ma se ne possono eliminare un bel po' durante il percorso.

Tipi di errori

L'arte dello sterminio degli errori è nota come *debugging*. Prima di poter eliminare un errore, dovete trovarlo. Nel caso di programmi piccoli, come quelli che visualizzano sullo schermo la scritta `Ciao, mondo!`, i luoghi in cui gli errori possono nascondersi sono limitati. Nei programmi di grandi dimensioni, invece, un errore può nascondersi ovunque e tentare di trovarlo può essere tanto frustrante quanto cercare l'ago in un pagliaio.

Per rendere più semplice la caccia agli errori, gli informatici hanno classificato gli errori in tre categorie.

- ✓ Errori di sintassi.
- ✓ Errori run-time.
- ✓ Errori logici.

Errori di sintassi

Un errore di *sintassi* si verifica quando il programmatore scrive un comando in modo errato. Se, per esempio, si digita **INTTEGER** invece di **INTEGER**, Visual Ba-

sic non sarà in grado di capire che cosa significa INTTEGER e si rifiuterà di eseguire la parte restante del programma.

Quando Visual Basic incontra un errore di sintassi, evidenzia la parola scritta male sullo schermo per mostrare esattamente dove si trova il problema. È sufficiente correggere la parola ed eseguire nuovamente il programma.

Persino un solo errore di sintassi impedirà al vostro programma di funzionare correttamente. Quando finalmente riuscirete a eseguire il programma per la prima volta, sarete certi che la vostra creazione non contiene errori di sintassi e quindi dovrete preoccuparvi soltanto degli errori run-time e degli errori logici.

Errori run-time

Un errore *run-time* si verifica quando un programma recupera dati che non sa come gestire. Un errore run-time è più infido di un errore di sintassi; il vostro programma può essere pieno zeppo di errori run-time, ma non ve ne accorgerete se prima non eseguite il programma.

Per simulare un errore run-time nella vostra vita, entrate in un fast-food Burghy e, quando la cameriera vi chiederà: "Che cosa desidera?", ordinate un'aragosta. Dato che la cameriera si aspetta un'ordinazione scelta dal menu di Burghy, non saprà come rispondere alla vostra richiesta e incapperà in un errore run-time.

Per un esempio di un errore run-time in un programma, si osservi la seguente formula scritta per calcolare un risultato:

```
AliquotaFiscale = TasseDovute / RedditoAnnuo
```

Di solito questa equazione funziona, a meno che `RedditoAnnuo` sia uguale a zero. Dato che è impossibile dividere qualsiasi numero per zero, se il reddito annuo è uguale a zero, il programma si interromperà.

Per scoprire un errore run-time, dovete provare il programma ipotizzando tutti i casi possibili: da qualcuno che preme un tasto sbagliato a qualche idiota che digita un numero negativo per la sua età.

Dato che il numero delle cose che possono andare male è praticamente infinito (legge di Murphy), potete capire perché qualsiasi programma contenga degli errori. Non è un pensiero confortante su cui meditare la prossima volta che viaggerete su un aereo controllato da un computer?

Errori logici

Il tipo di errore più difficile da individuare è un errore *logico*. Un errore logico si verifica quando il programma non funziona correttamente perché gli sono stati

dati comandi sbagliati oppure i comandi impartiti sono fuori fase rispetto ad altri comandi. Cosa? Com'è possibile che abbiate inserito dei comandi sbagliati se il programma l'avete scritto voi? Che ci crediate o meno, è semplice.

Chiunque stia educando degli adolescenti sa che, se si dice loro di falciare l'erba in giardino o di riassettare la loro stanza è possibile che lo facciano, ma non nel modo da voi desiderato. Invece di tagliare l'erba procedendo per file ben ordinate, un adolescente potrebbe procedere in cerchi e quindi lasciare il lavoro a metà. Oppure, invece di pulire la stanza raccogliendo gli abiti sporchi e gettando via le cartacce, un adolescente potrebbe accatastare tutto sotto il letto o nel corridoio.

In entrambi i casi, l'adolescente ha seguito le istruzioni ma queste ultime non erano abbastanza specifiche. Se un adolescente riesce a trovare un punto debole nelle vostre istruzioni lo sfrutterà; il computer si comporta allo stesso modo.

Poiché siete convinti di avere dato al computer i comandi giusti, non riuscirete a capire perché il programma non funziona. Ora sarete costretti a trovare il punto in cui le vostre istruzioni non sono abbastanza chiare. Se il programma è grande, questo potrebbe significare analizzare tutto il programma, riga dopo riga (com'è divertente programmare, vero?).

Caccia grossa agli errori

Fondamentalmente, per scovare ed eliminare gli errori presenti nel programma occorre seguire una procedura in quattro fasi.

1. **Capire che nel programma c'è un errore.**
2. **Trovare l'errore.**
3. **Trovare la causa dell'errore.**
4. **Eliminare l'errore.**

Capire che nel programma c'è un errore

Il metodo migliore per scoprire gli errori nel programma consiste nel farlo utilizzare da individui ignari, che nel mondo dei prodotti software commerciali vengono detti "clienti paganti".

Più persone proveranno il programma, maggiori saranno le probabilità di individuare gli errori di cui si ignorava l'esistenza. Gli errori possono essere evidenti come quelli che determinano un crash del computer oppure subdoli come quelli che arrotondano le cifre con il decimale sbagliato.

Dopo che si è concluso che nel programma c'è un errore, questo va rintracciato (gli ottimisti possono definire gli errori "funzioni non documentate").

Trovare l'errore

Trovare dove si nasconde un errore costituisce la parte più difficile. Il metodo più semplice (e più noioso) per trovare il nascondiglio di un errore consiste nell'eseguire il programma ed esaminarlo riga per riga; nel momento stesso in cui appare l'errore saprete con certezza quale riga l'ha causato.

Per programmi piccoli questo metodo è accettabile; per programmi grandi, è una vera follia.

Un'alternativa più rapida consiste nell'esaminare le parti del programma in cui pensate possa nascondersi l'errore. Se per esempio il vostro programma non stampa correttamente, l'errore potrebbe trovarsi nel codice BASIC che dice al computer come stampare.

Trovare la causa dell'errore

Quando avrete stabilito dove pensate che si nasconda, dovrete capire qual è la causa dell'errore.

Supponete che il programma debba stampare sullo schermo il vostro nome e stampi invece il numero della tessera della mutua; è possibile che il programma stampi tutto correttamente, ma recuperi il tipo di informazioni sbagliate da stampare.

Utilizzando le vostre incredibili capacità deduttive, capirete che l'errore (probabilmente) si trova nel punto in cui il programma cerca di stampare il vostro nome.

Eliminare l'errore

Dopo avere scoperto la causa del vostro errore dovrete correggere il programma, ma fate attenzione! Talvolta la correzione di un errore inserisce altri due o tre errori. Cosa? Come può succedere?

Confrontate questa situazione con quanto succede se c'è un problema nelle tubazioni di casa; la soluzione più semplice potrebbe essere quella di demolire una parete e inserire dei tubi nuovi. Questo potrebbe risolvere il problema delle tubazioni, tuttavia la demolizione della parete potrebbe anche provocare la rottura dei cavi elettrici presenti nel muro. Quindi avrete risolto i problemi delle tubazioni ma avrete anche creato un nuovo problema elettrico. Se si costruisce una nuova parete con i fili elettrici, si potrebbe per errore bloccare uno sfiato dell'aria condizionata centralizzata. Spostate la parete di 50 centimetri e il soffitto diventerà troppo debole al centro. Come potete vedere, il vostro piccolo problema nelle tubazioni si è moltiplicato.

Quindi bisogna fare attenzione quando si corregge un errore; talvolta è più semplice riscrivere una parte consistente di codice piuttosto che tentare di sistemare un errore all'interno del codice.

Il sistema migliore per evitare gli errori è non inserirli; come dire che per evitare i problemi di soldi basta averne sempre a sufficienza.

Dato che gli errori fanno la loro comparsa persino nei migliori programmi, al massimo si può sperare di ridurne il numero. Ecco alcuni consigli.

- Al fine di evitare gli errori, scrivete molti programmi piccoli e incollateli insieme per realizzare un unico grosso programma. Più piccoli saranno i programmi, più facile sarà isolare gli errori. In termini militari questa tattica viene definita *divide et impera*.
- Testate il programma ogni volta che lo si modifica: se il vostro programma funzionava bene prima che cambiaste le ultime tre righe, il problema potrà essere individuato in quelle tre righe.
- Trovate qualcuno da utilizzare come capro espiatorio: se il vostro programma si rifiuta di lavorare, prendetevela con vostra moglie, con il cane oppure con il destino. Questo non vi aiuterà a far funzionare nuovamente il programma ma vi farà sentire meglio per un paio di minuti.

Come Visual Basic rintraccia ed elimina gli errori

Visual Basic offre due sistemi principali per rintracciare ed eliminare gli errori: la progressione e l'osservazione.

- *Progressione* significa eseguire il programma riga per riga ed esaminare ogni istruzione. Dopo l'esecuzione di ogni riga, si osservi che cosa ha fatto il programma: se il programma funziona nel modo desiderato, la riga è a posto; in caso contrario, si è trovato un errore.
- L'*osservazione* permette di vedere quali dati vengono utilizzati dal programma in un determinato momento. Cercando dati specifici, come un nome o un numero telefonico, si può controllare se il programma sta memorizzando, stampando o modificando i dati in questione nella maniera corretta.

Analizzando un programma riga per riga e osservando quali dati vengono utilizzati è possibile trovare eventuali errori.

Procedere attraverso un programma riga per riga

Se non avete la più pallida idea di dove potrebbe nascondersi l'errore, dovrete esaminare tutto il programma riga per riga. A questo scopo, Visual Basic vi offre tre comandi.

- ✓ Debug-Esegui istruzione (premete F8).
- ✓ Debug-Esegui istruzione/routine (premete MAIUSC+F8).
- ✓ Debug-Esci da istruzione/routine (premete CTRL+MAIUSC+F8).

Il comando Esegui istruzione esegue tutto il programma una riga alla volta, comprese tutte le righe memorizzate in ciascuna procedura del programma.

Il comando Esegui istruzione/routine esegue tutto il programma, ma ogni volta che Visual Basic incontra una procedura salta le relative istruzioni.

Il comando Esci da istruzione/routine viene utilizzato insieme al comando Esegui istruzione. Quest'ultimo comando vi mostra, riga per riga, come sta funzionando una procedura; tuttavia, se decidete improvvisamente che non volete esaminare in questo modo il funzionamento di una procedura, utilizzate il comando Esci da istruzione/routine.

È possibile utilizzare insieme i due comandi in qualsiasi momento. Innanzitutto, utilizzate il comando Esegui istruzione per esaminare il programma riga per riga e poi utilizzate il comando Esci da istruzione/routine per uscire dalle procedure che il comando Esegui istruzione comincia a visualizzare. Infine, utilizzate il comando Esegui istruzione/routine per saltare le procedure che sapete che funzionano già correttamente.

Per utilizzare i comandi Esegui istruzione o Esegui istruzione/routine, si utilizzi questa procedura.

1. **Premete F8 oppure MAIUSC+F8 per selezionare rispettivamente Esegui istruzione oppure Esegui istruzione/routine.**

   Visual Basic visualizza una riga del vostro programma, come illustrato nella Figura 20.1

2. **Selezionate Fine dal menu Esegui quando volete fermarvi.**

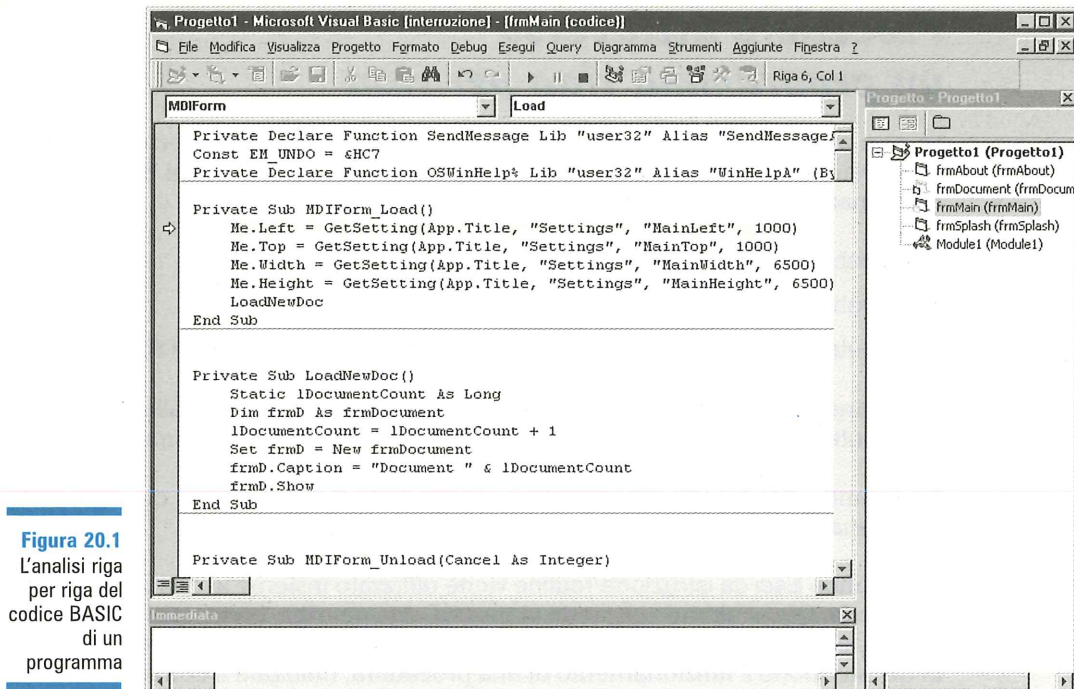



Figura 20.1 L'analisi riga per riga del codice BASIC di un programma

Esaminare le variabili

Analizzare il programma riga per riga è del tutto inutile se non sapete come il programma gestisce i dati. Per aiutarvi a vedere i valori contenuti da ogni variabile in un determinato momento, Visual Basic vi offre la finestra Controlla.

Questa finestra dice a Visual Basic: "Mostrami il contenuto delle variabili che ho deciso di esaminare mentre analizzo il programma riga per riga".

Per visualizzare la finestra Aggiungi espressione di controllo necessaria per determinare quali variabili si vogliono esaminare, utilizzate la procedura di seguito riportata.

1. **Premete F7, selezionate Visualizza-Codice oppure fate clic sull'icona Visualizza codice nella finestra Progetto per aprire la finestra Codice.**

2. **Evidenziate la variabile che volete esaminare e selezionate Debug-Aggiungi espressione di controllo.**

   Appare la finestra Aggiungi espressione di controllo, come illustrato nella Figura 20.2.

Mettete alla prova le vostre nuove conoscenze

1. **Che cos'è un bug (baco)?**

   a. Un errore che impedisce al programma di funzionare correttamente.

   b. Un'orribile, piccola creatura con sei o più gambe e una dura corazza che scricchiola quando viene calpestata.

   c. Una farfalla che si suicida infilandosi nel vostro computer.

4. Qualcosa che i ragazzini utilizzano per spaventare le bambine.

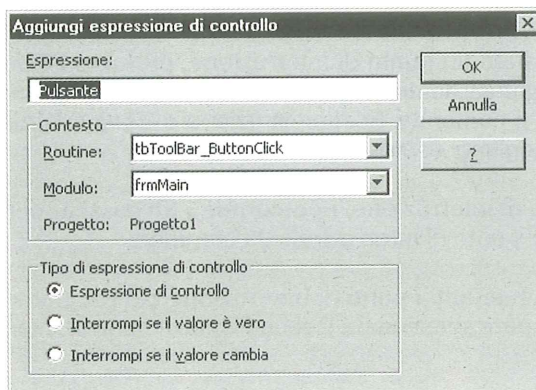
2. **In che modo Visual Basic rintraccia gli errori?**

   a. Arrivando dotato di bachi propri.

   b. Con la colla e le apposite esche.

   c. Rendendo la programmazione tanto difficile da impedire a chiunque di scrivere un baco anche se volesse.

   d. Analizzando il programma e osservando se i dati vengono gestiti correttamente.

Figura 20.2 La finestra di dialogo Aggiungi espressione di controllo per il debugging del programma

3. **Fate clic su OK.**

4. **Premete F8 o MAIUSC+F8 per selezionare il comando. Premete F8 oppure MAIUSC+F8 per selezionare i comandi Esegui istruzione (se desiderate esaminare i meandri di una procedura) oppure Esegui istruzione/routine (se desiderate saltare il codice racchiuso in una procedura).**

   La finestra Aggiungi espressione di controllo visualizza il valore della vostra variabile ogni volta che selezionate il comando Esegui istruzione o Esegui istruzione/routine.

5. **Selezionate Fine dal menu Esegui oppure fate clic sull'icona Fine nella barra degli strumenti quando volete fermarvi.**

Impostare i punti di interruzione

Sia il comando Esegui istruzione che il comando Esegui istruzione/routine cominciano dall'inizio del programma e continuano fino alla fine. Questo metodo è accettabile per i programmi di piccole dimensioni ma diventa insopportabile per i programmi di ampie dimensioni.

Per saltare ampie sezioni del vostro programma che sapete (o sperate) funzionare correttamente, potete impostare un punto di interruzione. Un *punto di interruzione* dice a Visual Basic: "Esegui il programma fino a quando mi raggiungi, poi attendi che io dia il comando Esegui istruzione, Esegui istruzione/routine oppure Esegui".

Per impostare un punto di interruzione, si utilizzi questa procedura.

1. **Premete F7, selezionate Visualizza-Codice oppure fate clic sull'icona Visualizza codice nella finestra Progetto per aprire la finestra Codice.**
2. **Fate clic sulla riga nella quale volete impostare il punto di interruzione.**
3. **Premete F9 oppure selezionate Debug-Imposta-rimuovi punto di interruzione.**

Dopo che avrete impostato un punto di interruzione, premete F5 per eseguire il programma fino al raggiungimento del punto di interruzione (Figura 20.3). Dopodiché si può utilizzare il comando Esegui istruzione o Esegui istruzione/routine insieme al comando Aggiungi espressione di controllo.

Per eliminare un punto di interruzione, ripercorrete i tre passi appena elencati e fate clic sulla riga con il punto di interruzione da eliminare.

Per eliminare velocemente tutti i punti di interruzione dal programma, premete CTRL+MAIUSC+F9 oppure selezionate Debug-Rimuovi punti di interruzione.

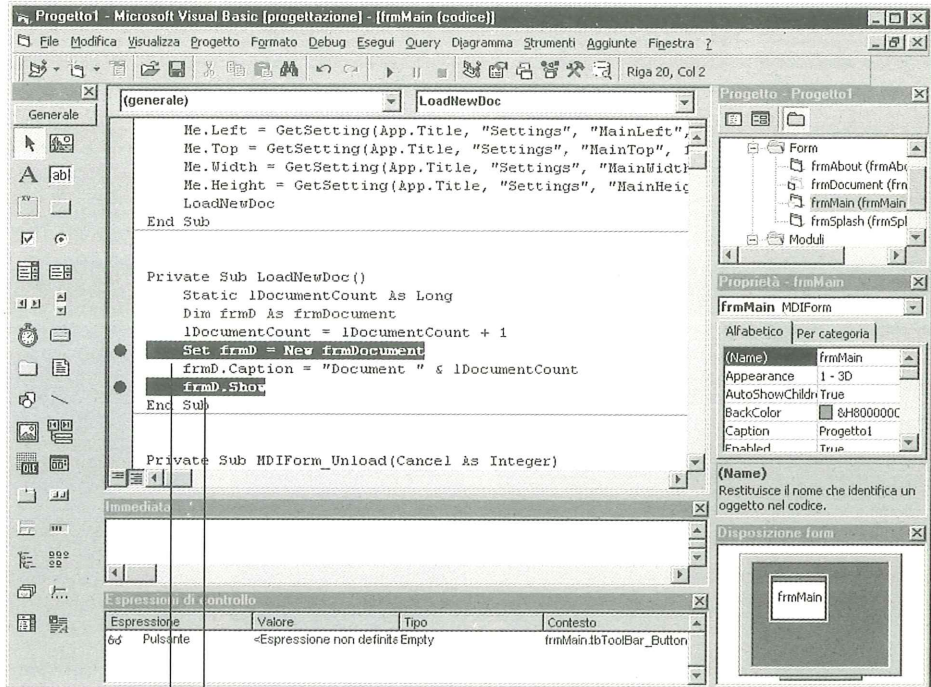
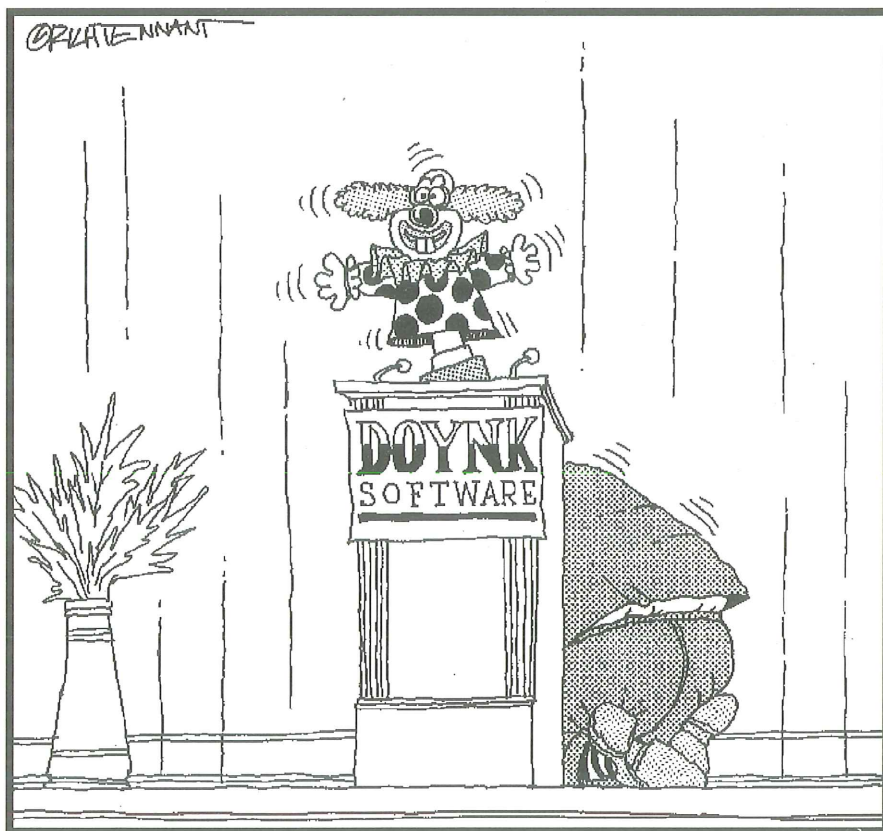



Figura 20.3 L'utilizzo dei punti di interruzione nella finestra Codice

Parte V
Prendere decisioni

"QUALCUNO HA DETTO CHE SIAMO IMPAZZITI NEL TENTATIVO DI LANCIARE SUL MERCATO QUESTO PRODOTTO IN TEMPO ..."

In questa parte...

I programmi contengono semplicemente le istruzioni che dicono al computer che cosa fare. Il programma più semplice si limita a includere un elenco consistente di istruzioni che il computer esegue, una dopo l'altra.

Tuttavia l'esecuzione sequenziale delle istruzioni non costituisce una garanzia di programma utile. La maggior parte dei programmi deve ricevere i dati e poi decidere come utilizzarli.

Queste capacità decisionali fanno sembrare i computer capaci di reagire, vivi e intelligenti (o almeno vivi). Quando un programma sarà in grado di dire al computer come prendere delle decisioni, allora potrà cominciare a fare qualcosa di utile.

Capitolo 21

Le istruzioni If-Then e If-Then-Else

In questo capitolo

- Specificare una condizione
- Utilizzare le istruzioni If-Then e If-Then-End If
- Utilizzare le istruzioni If-Then-Else e If-Then-ElseIf

Tutti hanno preso decisioni in base a determinate condizioni, come: "Non preoccuparti tesoro. Se la partita di calcio finirà presto, falcerò l'erba del prato". Oppure: "Se tu smettessi di rimpinzare il gatto, forse smetterebbe di ingrassare". Visual Basic non è diverso quando si tratta di prendere delle decisioni: controlla determinate condizioni e poi reagisce.

Condizioni

Una *condizione* deve rappresentare un valore che sia True o False. Le condizioni possono essere le seguenti.

- ✓ Una variabile singola.
- ✓ Un'espressione.

Se una condizione è rappresentata da una variabile singola, allora questa deve possedere un valore che sia True o False. È possibile controllare il valore di una variabile in due modi; il primo consiste nel controllare in modo specifico se una variabile è uguale a True, come nel seguente esempio:

```
If TroppoCaldo = True Then
```

Il secondo (che è anche più breve da scrivere) vi consente di eliminare `= True` e di scrivere semplicemente:

```
If TroppoCaldo Then
```

Non è necessario specificare se questa variabile debba essere uguale a True, dato che Visual Basic controllerà ugualmente se essa è True o False.

Se volete controllare in modo specifico se una variabile è False, potete procedere così:

```
If TroppoFreddo = False Then
```

Come scorciatoia, eliminate `= False` e scrivete semplicemente:

```
If Not TroppoFreddo Then
```

Se una condizione è costituita da un'espressione, l'intera espressione dovrà rappresentare un valore che sia True o False, come:

```
If Età >= 21 Then
```

In questo esempio, la condizione è Età >= 21

In questa condizione, se il valore di `Età` è superiore o uguale a 21, l'espressione sarà True, altrimenti sarà False.

Nell'esempio di seguito riportato, se la stringa memorizzata nella proprietà `Text` di una casella di testo denominata `txtNome` contiene la stringa "Capitano Nando", l'espressione è True; altrimenti è False.

```
If (txtNome.Text = "Capitano Nando") Then
```

In questa istruzione, le parentesi non hanno alcuna influenza sul codice ma servono semplicemente a identificare la condizione.

Nel seguente esempio, l'espressione `GattoPresente And GattoMonello` dà come risultato False:

```
GattoPresente = False
GattoMonello = True
If GattoPresente And GattoMonello Then
```

Per rendere la condizione definita da un'istruzione If-Then più facilmente individuabile, si può racchiudere tra parentesi, come viene illustrato nella seguente riga di codice:

```
If (GattoPresente And GattoMonello) Then
```

Ora che si è chiarito che cosa sono e come funzionano le condizioni, vediamo come inserirle in un'istruzione If-Then (paragrafo successivo).

L'istruzione If-Then

Per prendere delle decisioni, Visual Basic utilizza le istruzioni If-Then. Un'istruzione If-Then è il metodo utilizzato da Visual Basic per verificare se una condizione è True o False.

Se la condizione è True, Visual Basic segue una certa istruzione; se la condizione è False, Visual Basic ignora questa istruzione.

Tutto questo potrebbe sembrare piuttosto prolisso, ecco quindi la versione condensata dell'istruzione:

```
If Condizione Then Istruzione
```

In sostanza questo codice dice a Visual Basic che se una determinata condizione è True, allora deve obbedire all'istruzione che segue.

A prescindere dalla condizione, il valore ottenuto deve sempre essere True o False.

Vediamo alcuni esempi.

```
If Numero > 25 Then txtNota.Text = "Pieno"
```

Ecco come Visual Basic interpreta questo codice.

1. Questo comando dice: "Controlla una variabile denominata `Numero` e vedi se il suo valore è superiore a `25`. In caso affermativo, inserisci la stringa `"Pieno"` nella proprietà `Text` di una casella di testo denominata `txtNota`".
2. "Se il valore della variabile `Numero` è uguale o inferiore a `25`, passa all'istruzione successiva nel programma Visual Basic".

```
If Affamato OR Annoiato Then Messaggio = "Mangiamo."
```

Ecco come Visual Basic interpreta questo codice.

1. Questo comando dice: "Controlla il valore di una variabile denominata `Affamato` e verifica il valore di una variabile denominata `Annoiato`. Se una o l'altra hanno un valore uguale a True, allora crea una variabile denominata `Messaggio` e assegnale la stringa `"Mangiamo"`.
2. "Se entrambe le variabili sono False, passa all'istruzione successiva nel programma Visual Basic".

Una tipica istruzione If-Then controlla se una determinata istruzione è True o False e quindi esegue una singola istruzione. Ma che cosa succede se si vuole verificare una condizione e quindi fare in modo che il computer esegua due o

più istruzioni? In questo caso, dovete utilizzare una forma diversa di istruzione If-Then, denominata istruzione If-Then-End If.

L'istruzione If-Then-End If

L'istruzione If-Then-End If permette a Visual Basic di verificare una singola condizione; e se questa è True, il programma eseguirà una serie di istruzioni. Ecco la sintassi corretta di If-Then-End If:

```
If Condizione Then
  Istruzione1
  Istruzione2
End If
```

In sostanza questo codice dice a Visual Basic: "Verifica una condizione; se è True, allora esegui tutte le istruzioni successive fino al raggiungimento di `End If`".

Ecco un esempio semplicissimo:

```
If Black_Out = True Then
  Accendi_Candele = True
  txtAvvertimento.Text = "Hai appena perso tutto il tuo lavoro"
End If
```

Ecco come Visual Basic interpreta questo codice.

1. La prima riga dice: "Controlla il valore di una variabile denominata `Black_Out`; se è True esegui le due istruzioni successive, se è False non fare nulla".
2. "Attribuisci il valore `True` alla variabile `Accendi_Candele`".
3. "Attribuisci la stringa `"Hai appena perso tutto il tuo lavoro"` alla proprietà `Text` della casella di testo `txtAvvertimento`".

L'istruzione If-Then-Else

L'istruzione If-Then offre al programma la capacità di prendere decisioni in base a determinate condizioni. Se questa possibilità non rappresenta l'apice della vostra carriera di programmatore, allora preparatevi a conoscere l'istruzione If-Then-Else.

Il problema di un'istruzione If-Then o If-Then-End If è che potreste avere bisogno di una quantità eccessiva di istruzioni per verificare sia le condizioni True che le condizioni False.

Esiste un sistema più semplice? La risposta, ovviamente, è sì! Visual Basic vi offre l'istruzione If-Then-Else.

L'istruzione If-Then-Else più semplice assomiglia alla seguente:

```
If Condizione Then
  IstruzioniVero
Else
  IstruzioniFalso
End If
```

Questa istruzione dice a Visual Basic: "Se la condizione è True, esegui la prima serie di istruzioni; se la condizione è False, esegui la seconda serie di istruzioni".

Mettete alla prova le vostre nuove conoscenze

1. Che cos'è una condizione?

a. Qualcosa che vi fa venir voglia di grattarvi la testa o le dita dei piedi.

b. Una scusa che utilizzate per non fare qualcos'altro, come "Oggi il mio cuore è davvero in cattive condizioni, penso che starò in casa tutto il giorno a guardare la TV".

c. Un sistema utilizzato dai vostri genitori per minacciarvi. "Se non ti comporti bene, andrai a letto senza cena".

d. Una variabile o un'espressione che verifica se il valore è True o False.

2. Qual è l'alternativa alla seguente condizione?

`If SenzaSoldi = False Then`

a. `If SenzaSoldi Then "Lamentarsi per la concorrenza degli altri paesi."`

b. `If SenzaSoldi Then "Eleggere un nuovo Governo."`

c. `If Not SenzaSoldi Then`

d. `If SenzaSoldi = True Or False Then`

Quindi come si possono modificare le seguenti istruzioni?

```
If Giorno > 15 Then txtLeggimi.Text = "Le bollette sono
scadute!"
If Giorno <= 15 Then txtLeggimi.Text = "Paga le bollette!"
```

A seconda della condizione utilizzata, si riscriveranno queste istruzioni in due modi. Se viene utilizzata la condizione `Giorno > 15`, si ottiene quanto segue alla pagina successiva.

```
If Giorno > 15 Then
  txtLeggimi.Text = "Le bollette sono scadute!"
Else
  txtLeggimi.Text = "Paga le bollette!"
End If
```

Ma se viene utilizzata la condizione `Giorno <= 15`, si ottiene invece:

```
If Giorno <= 15 Then
  txtLeggimi.Text = "Paga le bollette!"
Else
  txtLeggimi.Text = "Le bollette sono scadute!"
End If
```

Entrambi i tipi di istruzione If-Then-Else sono perfettamente accettabili: è solo una questione di preferenza personale.

È possibile inserire tutte le istruzioni desiderate tra le righe If-Then-Else e tra le righe Else ed End If.

Uno svantaggio possibile delle istruzioni If-Then-Else sta nel fatto che se la prima condizione è False, Visual Basic esegue ciecamente il secondo gruppo di istruzioni. Se non si vuole che questo succeda, andrà specificata una condizione per la seconda serie di istruzioni. A questo scopo dovrà essere utilizzata l'istruzione If-Then-ElseIf.

L'istruzione If-Then-ElseIf

Un'istruzione If-Then-ElseIf assomiglia a quanto segue:

```
If Condizione1 True
  Istruzioni1
ElseIf Condizione2 Then
  Istruzioni2
End If
```

Questo codice dice a Visual Basic: "Se `Condizione1` è True, allora esegui la prima serie di istruzioni. Se invece `Condizione1` è False, verifica se `Condizione2` è True. In caso affermativo, esegui la seconda serie di istruzioni; se `Condizione2` è False non fare nulla".

Con le istruzioni If-Then-Else, il computer esegue sempre almeno una serie di istruzioni. Invece con l'istruzione If-Then-ElseIf è possibile che il computer non esegua alcuna istruzione, proprio come un adolescente ribelle.

Per esempio:

```
If Giorno > 15 Then
  txtLeggimi.Text = "Le bollette sono scadute!"
ElseIf Giorno > 10 Then
  txtLeggimi.Text = "Paga le bollette!"
End If
```

Che cosa succede se il valore di `Giorno` è 12?

1. Visual Basic controlla la prima condizione e conclude che l'espressione `12 > 15` è False (dato che il valore di `Giorno` è 12).
2. Poi Visual Basic controlla la seconda condizione e conclude che l'espressione `12 > 10` è True e, di conseguenza, attribuisce la stringa `"Paga le bollette"` alla proprietà `Text` di una casella di testo denominata `txt.Leggimi`.

Ecco la parte difficile. Che cosa succede se il valore di `Giorno` è 6?

1. Visual Basic controlla la prima condizione e conclude che l'espressione `6 > 15` è False e quindi ignora la prima serie di istruzioni.
2. Poi Visual Basic controlla la seconda condizione e conclude che l'espressione `6 > 10` è False e quindi ignora la seconda serie di istruzioni.
3. Infine Visual Basic raggiunge la fine dell'istruzione If-Then-ElseIf. Dato che nessuna delle istruzioni era True, non è stata eseguita alcuna istruzione.

Per gestire numerose possibilità, dovete utilizzare parecchie condizioni ElseIf.

Diverse opzioni con If-Then-ElseIf

Per verificare numerose condizioni, utilizzate parecchie condizioni ElseIf, come riportato di seguito:

```
If Condizione1 Then
  Istruzioni1
ElseIf Condizione2 Then
  Istruzioni2
ElseIf Condizione3 Then
  Istruzioni3
End If
```

Questo codice dice a Visual Basic: "Se `Condizione1` è True, esegui `Istruzioni1`. Se invece `Condizione1` è False, verifica se `Condizione2` è True. Se `Condizione2` è True, esegui `Istruzioni2`. Se `Condizione1` è False e `Condizione2` è False, controlla se `Condizione3` è True. Se `Condizione3` è True, allora esegui `Istruzioni3`".

Ancora una volta, è possibile che tutte le condizioni siano False e che il computer non esegua alcuna istruzione.

Si possono utilizzare tutte le righe ElseIf desiderate. Ovviamente, più se ne utilizzano, più l'istruzione If-Then-ElseIf diventerà complessa. ("Che cosa succede se la Condizione3 è False, ma la Condizione4 è True? Aspettate un attimo, che cosa doveva succedere?").

Accertarsi che il computer esegua almeno una serie di istruzioni

È possibile passare attraverso una struttura If-Then-ElseIf senza che il computer esegua neppure un'istruzione. Per accertarsi che il computer esegua almeno una serie di istruzioni, è necessario aggiungere l'istruzione Else alla fine, come illustrato nel seguente esempio:

```
If Condizione1 Then
  Istruzioni1
ElseIf Condizione2 Then
  Istruzioni2
Else
  IstruzioniDefault
End If
```

Questo codice dice a Visual Basic: "Se Condizione1 è True, esegui la prima serie di istruzioni. Se invece Condizione1 è False, verifica il valore di Condizione2. Se Condizione2 è True, esegui la seconda serie di istruzioni. Se tutte le condizioni sono False, allora procedi ed esegui l'ultima serie di istruzioni".

Nidificare le istruzioni If-Then

Se si vuole è possibile racchiudere numerose istruzioni If-Then una nell'altra, nel seguente modo:

```
If Età > 21 Then
  If Valutazione = 10 Then
     txtAzione.Text = "Chiedi un appuntamento."
  End If
Else
  txtAzione.Text = "Mi dispiace, sei troppo giovane."
End If
```

Se il valore di `Età` fosse 23 e il valore di `Valutazione` fosse 10, Visual Basic interpreterebbe questo codice come segue.

1. Visual Basic verificherebbe la prima condizione e concluderebbe che l'espressione `Età > 21` è True (dato che il valore di `Età` è 23).
2. Dopodiché, Visual Basic verificherebbe la seconda condizione e concluderebbe che l'espressione `Valutazione = 10` è True e quindi attribuirebbe la stringa `"Chiedi un appuntamento."` alla proprietà `Text` di una casella di testo denominata `txtAzione`.

Se il valore di `Età` fosse 23 ma il valore di `Valutazione` fosse solo 9, Visual Basic interpreterebbe questo codice come segue.

1. Visual Basic verificherebbe la prima condizione e concluderebbe che l'espressione `Età > 21` è True (dato che il valore di `Età` è 23).
2. Dopodiché, Visual Basic verificherebbe la seconda condizione e concluderebbe che l'espressione `Valutazione = 10` è False e quindi non succederebbe alcunché.

Infine, se il valore di `Età` fosse 13 e il valore di `Valutazione` fosse 10, Visual Basic interpreterebbe questo codice come segue.

1. Visual Basic verificherebbe la prima condizione e concluderebbe che l'espressione `Età > 21` è False (dato che il valore di `Età` è 13).
2. Visual Basic passerebbe alla parte Else dell'istruzione If-Then-ElseIf e inserirebbe la stringa `"Mi spiace, sei troppo giovane."` nella proprietà `Text` della casella di testo denominata `txtAzione`. Notate che in questo caso il valore di `Valutazione` è del tutto irrilevante.

Fate attenzione quando nidificate le istruzioni If-Then una nell'altra, dato che le istruzioni If-Then nidificate potrebbero comportarsi in modo inaspettato. Per esempio, nel codice precedente, se il valore di `Età` fosse 23 ma il valore di `Valutazione` fosse solo 9, potreste rimanere sorpresi nel notare che questo codice non inserirebbe alcuna stringa nella proprietà `Text` della casella di testo `txtAzione`.

Capitolo 22

Select Case e le strutture di controllo nidificate

In questo capitolo

- Esaminare le istruzioni Select Case
- Accertarsi che il computer esegua almeno una serie di istruzioni
- Utilizzare le strutture di controllo nidificate

Il problema principale che si verifica con l'utilizzo di molte istruzioni If-Then-ElseIf, discusse nel Capitolo 21, è che queste istruzioni sono brutte, difficili da leggere e da capire e noiose da scrivere.

Osservate quanto segue:

```
If Chiamante = "Franco" Then
  txtRisposta.Text = "Sì"
ElseIf Chiamante = "Matteo" Then
  txtRisposta.Text = "OK, ma solo se paghi tu"
ElseIf Chiamante = "Gianni" Then
  txtRisposta.Text = "Stasera devo lavarmi i capelli"
ElseIf Chiamante = "Stefano" Then
  txtRisposta.Text = "Questo è un messaggio registrato"
End If
```

Qual è l'alternativa a una proliferazione infinita di ElseIf, che può essere motivo di confusione?

Un'alternativa potrebbe essere gettare la vostra copia di Visual Basic fuori dalla finestra e trovare qualcuno che scriva i programmi al posto vostro, ma è decisamente più pratico utilizzare la cosiddetta istruzione Select Case.

L'istruzione Select Case

L'istruzione Select Case ha il seguente aspetto:

```
Select Case NomeVariabile
  Case X
     Istruzioni1
  Case Y
     Istruzioni2
  Case Z
     Istruzioni3
End Select
```

Questa istruzione dice a Visual Basic: "Osserva il valore della variabile denominata `NomeVariabile`. Se questo valore è uguale a `X`, esegui `Istruzioni1`; se questo valore è uguale a `Y`, esegui `Istruzioni2`; se questo valore è uguale a `Z`, esegui `Istruzioni3`".

Sostituendo le numerose istruzioni If-Then-ElseIf riportate all'inizio di questo capitolo con l'istruzione Select Case, si otterrebbe quanto segue:

```
Select Case Chiamante
  Case "Franco"
     txtRisposta.Text = "Sì"
  Case "Matteo"
     txtRisposta.Text = "OK, ma solo se paghi tu"
  Case "Gianni"
     txtRisposta.Text = "Stasera devo lavarmi i capelli"
  Case "Stefano"
     txtRisposta.Text = "Questo è un messaggio registrato"
End Select
```

Si noti l'aspetto più ordinato e l'eliminazione delle ripetizioni dei comandi ElseIf e Then.

In base al numero dei valori da controllare, si possono inserire tutte le righe Case desiderate in un'istruzione Select Case.

Utilizzare l'istruzione Select Case con gli operatori di confronto

Di solito, l'istruzione Select Case richiede il valore preciso da esaminare, ma utilizzando gli operatori di confronto quali <, <= o <>, è possibile fare in modo che l'istruzione Select Case controlli se una variabile rientra in un determinato intervallo di valori.

Per fare in modo che l'istruzione Select Case utilizzi gli operatori di confronto, bisogna utilizzare la parola magica riservata `is`. Di conseguenza, questa istruzione Select Case:

```
Select Case Giorno
  Case is > 15
     txtLeggimi.Text = "Le bollette sono scadute!"
  Case is <= 15
     txtLeggimi.Text = "Paga le bollette!"
End Select
```

equivale alla seguente istruzione If-Then:

```
If Giorno > 15 Then
  txtLeggimi.Text = "Le bollette sono scadute!"
ElseIf Giorno < 15 Then
  txtLeggimi.Text = "Paga le bollette!"
End If
```

Verificare che il computer esegua almeno una serie di istruzioni

In modo analogo all'istruzione If-Then-ElseIf, è possibile che il computer non esegua alcuna istruzione all'interno di un'istruzione Select Case. Per accettarsi che il computer esegua almeno una serie di istruzioni, si deve utilizzare nuovamente il comando magico Else. Esaminate l'esempio seguente.

```
Select Case Giorno
  Case 1
     Istruzioni1
  Case 2
     Istruzioni2
  Case 3
     Istruzioni3
  Case Else
     IstruzioniDiDefault
End Select
```

Il codice sopra riportato dice a Visual Basic: "Se il valore di `Giorno` è uguale a `1`, esegui la prima serie di istruzioni; se il valore di `Giorno` è uguale a `2`, esegui la seconda serie di istruzioni; se il valore di `Giorno` è uguale a `3`, esegui la terza serie di istruzioni. Se il valore di `Giorno` non è uguale né a `1`, né a `2`, né a `3`, allora esegui l'ultima serie di istruzioni.

Strutture di controllo nidificate

Uno dei giocattoli più semplici che divertono i bambini senza mai annoiarli sono le scatolette cinesi infilate una dentro l'altra. Ogni volta che aprite una scatola ne trovate una più piccola all'interno; alla fine raggiungete un punto in cui non vi sono più scatole da aprire per cui siete costretti a fermarvi.

Di solito una normale istruzione Select Case può contenere uno o più gruppi di istruzioni, come viene illustrato nel seguente esempio:

```
Select Case ID
  Case 123
     chkFranco.Value = True
  Case 124
     chkMario.Value = True
  Case 125
     chkMarta.Value = True
End Select
```

Invece di inserire istruzioni noiose in un'istruzione Select Case, è possibile scrivere istruzioni If-Then e Select Case all'interno di un'istruzione Select Case, come viene di seguito illustrato:

```
Select Case QI
  Case 120
     Select Case Età
        Case < = 9
           txtAnalisi.Text = "Devi essere un bambino
intelligente."
     End Select
End Select
```

Ecco come Visual Basic interpreta questo codice.

1. La prima riga dice: "Controlla il valore memorizzato nella variabile `QI`, quindi continua con la seconda riga".
2. La seconda riga dice: "Se il valore di `QI` è esattamente uguale a `120` continua con la terza riga; se il valore di `QI` è diverso (come 119, 121 oppure 3) passa alla settima riga".
3. La terza riga dice: "Controlla il valore memorizzato in una variabile denominata `Età` e poi continua alla quarta riga".
4. La quarta riga dice: "Se il valore memorizzato nella variabile `Età` è `9`, continua con la quinta riga; se il valore di `Età` è maggiore di 9 (come 13, 86 oppure 10) passa alla sesta riga".

5. La quinta riga dice: "Attribuisci la stringa "Devi essere un bambino intelligente." alla proprietà Text della casella di testo txtAnalisi".
6. La sesta riga dice: "Questa è la fine di un'istruzione Select Case".
7. La settima riga dice: "Questa è la fine di un'altra istruzione Select Case".

Potete inserire un'istruzione If-Then all'interno di un'istruzione Select Case oppure un'istruzione Select Case all'interno di un'istruzione If-Then (divertente, vero?).

Sebbene non esista un limite teorico al numero di istruzioni If-Then o Select Case che possono essere sistemate le une nelle altre, meno se ne impiegano, più semplice il codice risulta da capire. La regola pratica generale sostiene che se avete nidificato più di tre istruzioni If-Then o Select Case l'una nell'altra, probabilmente non sapevate quello che stavate facendo.

Quando si nidificano più istruzioni, sarebbe meglio applicare un rientro alle istruzioni in modo che sia più semplice capire dove cominciano e dove finiscono. Per esempio, si noti quanto il programma seguente sembri confuso senza rientri:

```
Select Case Stipendio
Case 1200
If Nome ="Roberto" Then
txtRivedi.Text = "Nessun aumento quest'anno, ha, ha, ha!"
ElseIf Nome "Sofia" Then
txtRivedi.Text = "D'accordo, cosa ne dici di un aumento del
5%?"
End If
End Select
```

Ecco lo stesso listato con i rientri:

```
Select Case Stipendio
  Case 1200
     If Nome ="Roberto" Then
          txtRivedi.Text = "Nessun aumento quest'anno, ha,
ha, ha!"
     ElseIf Nome "Sofia" Then
          txtRivedi.Text = "D'accordo, cosa ne dici di un
aumento del 5%?"
     End If
End Select
```

Dal punto di vista del computer, entrambi i programmi sono identici, ma dal punto di vista del programmatore, il secondo, quello con i rientri, è molto più semplice da leggere e da capire.

Mettete alla prova le vostre nuove conoscenze

1. **Quante strutture di controllo (istruzioni If-Then o Select Case) è possibile nidificare?**

   a. Il limite è determinato dalle restrizioni imposte dal Governo.

   b. Il limite è determinato dalle applicazioni teoriche che fanno seguito alle implicazioni della Teoria della Relatività di Einstein, reinterpretate da un avvocato.

   c. Il limite è 55. Se lo superate rischiate di essere fermati da una pattuglia della polizia stradale.

   d. Non esiste un limite. Tuttavia, in presenza di una quantità eccessiva di strutture di controllo nidificate, il vostro programma diventerà più difficile da leggere e da capire.

2. **Che cosa si può fare per rendere le strutture di controllo nidificate più semplici da leggere e da capire?**

   a. Evitare di utilizzare le strutture di controllo nidificate.

   b. Evitare di programmare.

   c. Limitare il numero di strutture di controllo nidificate e utilizzare i rientri per rendere facilmente individuabili le istruzioni If-Then o Select Case.

   d. Stampare con caratteri grandi in grassetto e utilizzare istruzioni brevi: "Guarda Nino che corre. Nino corre veloce".

Parte VI

Entrare in un ciclo

I VERI PROGRAMMATORI ODIANO IL CAFFÈ DECAFFEINATO

In questa parte...

Ciascun programma contiene istruzioni che dicono al computer che cosa fare. A volte queste istruzioni vengono utilizzate un'unica volta, come nel caso delle istruzioni che dicono al computer che cosa fare quando viene caricato il programma.

Altre volte determinate istruzioni possono essere utilizzate ripetutamente. Invece di scrivere queste istruzioni ogni volta, i programmatori hanno inventato qualcosa di magico definito *ciclo*.

In pratica, un ciclo dice al computer: "Vedi quelle istruzioni laggiù? Ripetile un certo numero di volte e poi fermati." Grazie ai cicli, i programmatori non sono obbligati a scrivere un'infinità di volte le stesse istruzioni, risparmiando così molto tempo, pratica in cui questi individui se la cavano egregiamente.

Capitolo 23

I cicli Do While e Do-Loop While

In questo capitolo

- Utilizzare il ciclo Do While
- Determinare quante volte viene eseguito il ciclo Do While
- Come funziona il ciclo Do-Loop While e quando utilizzarlo

I cicli Do While non fanno nulla senza prima controllare se una determinata condizione è vera. In caso affermativo il ciclo procede e fa qualcosa, altrimenti passa alla successiva istruzione nel programma Visual Basic".

È possibile individuare dei cicli Do While nelle esperienze di tutti i giorni, come quando dei colleghi si dicono: "Continua a infilare roba nella mia valigetta finché nessuno ci guarda. Non appena qualcuno gira lo sguardo verso di me, fermati e fai altro".

Un ciclo Do While assomiglia a quanto segue:

```
Do While Condizione
  Istruzioni
Loop
```

La condizione deve essere una variabile o un'espressione che rappresenti un valore True o False. Un ciclo Do While può contenere una o più istruzioni.

Come funziona il ciclo Do While

La prima volta che Visual Basic vede un ciclo Do While dice: "OK, il valore della condizione è True o False? Se è False devo ignorare tutte le istruzioni all'interno del ciclo Do While. Se è True, devo eseguire tutte le istruzioni contenute nel ciclo Do While".

Per esempio, il seguente codice comprende un ciclo Do While:

```
Contatore = 0
Do While Contatore <> 5
  Contatore = Contatore + 1
  txtCountatore. Testo = CStr(Contatore)
Loop
```

Ecco che cosa significa questo codice.

1. La prima riga dice: "Attribuisci il valore `0` a una variabile denominata `Contatore`".
2. La seconda riga dice: "Finché il valore di `Contatore` non è uguale a 5, continua a ripetere tutte le istruzioni racchiuse tra la riga `Do While` e la riga `Loop`".
3. La terza riga dice " Aggiungi 1 (uno) al valore della variabile Contatore.
4. La quarta riga dice: "Prendi il valore di `Contatore` e convertilo dentro una stringa e inseriscilo dentro il Testo proprietà di una casella di testo chiamata txtContatore"
5. La quarta riga dice: "Questa è la fine del ciclo Do While. Torna indietro alla seconda riga dove comincia il ciclo Do While, ma solo se il valore di `Contatore <> 5`" è True.

Ogni volta che il ciclo viene eseguito, il valore di `Contatore` viene aumentato di 1. Non appena il valore di `Contatore` diviene uguale a 5, la condizione `Contatore <> 5` diviene False e il ciclo Do While si interrompe.

Quante volte viene ripetuto il ciclo Do While?

Se la condizione di un ciclo Do While è False, nessuna delle due istruzioni all'interno del ciclo Do While verrà eseguita. In questo caso, il ciclo Do While non verrà ripetuto.

Se la condizione di un ciclo Do While è True, allora il ciclo verrà ripetuto almeno una volta.

Se la condizione di un ciclo Do While è sempre True, allora il ciclo verrà ripetuto un numero infinito di volte, fino a quando spegnerete il computer oppure l'universo esploderà con un altro Big Bang.

Quando un ciclo si ripete all'infinito, viene detto *ciclo senza fine* (endless loop).

I cicli infiniti faranno entrare il programma in un circolo vizioso e gli impediranno di funzionare nella maniera corretta (dato che il ciclo non si interrompe mai). Per evitare un ciclo infinito, è necessario assicurarsi che almeno un'istruzione all'interno del ciclo Do While modifichi il valore True o False della condizione che controlla il ciclo.

Quando utilizzare un ciclo Do While

Un ciclo Do While va utilizzato ogni volta che si ha bisogno di fare quanto segue.

- ✓ Ripetere qualcosa zero o più volte.
- ✓ Continuare a ripetere qualcosa fino a quando una determinata condizione resta True.

Come funziona il ciclo Do-Loop While

I cicli Do-Loop in sostanza dicono a Visual Basic: "Procedi e fai qualcosa fino a che una determinata condizione ti dirà di fermarti."

È possibile individuare i cicli Do-Loop While nella vita di tutti i giorni, come quando i genitori dicono ai propri figli: "Arrangiati e fai quello che vuoi basta che non mi disturbi".

Un ciclo Do-Loop While assomiglia a quanto segue:

```
Do
  Istruzioni
Loop While Condizione
```

La condizione deve essere una variabile o un'espressione che rappresenti un valore True o False. Un ciclo Do-Loop While può contenere una o più istruzioni.

La prima volta che Visual Basic vede un ciclo Do-Loop While dice: "Per prima cosa eseguo tutte le istruzioni presenti nel ciclo e poi controllo se il valore della condizione è True o False; se è False mi fermo, se è True dovrò ripetere ancora una volta tutte le istruzioni del ciclo Do-Loop While".

Per esempio, il seguente è un tipico ciclo Do-Loop While in grado di contare:

```
Contatore = 0
Do
  Contatore = Contatore + 1
Loop While Contatore < 5
```

Ecco che cosa significa questo codice.

1. La prima riga dice: “Crea una variabile denominata `Contatore` e imposta a `0` il suo valore”.
2. La seconda riga dice: “Questo è l’inizio di un ciclo Do-Loop While”.
3. La terza riga dice: “Prendi il valore di `Contatore` e aggiungi 1. Ora memorizza questo valore nella variabile `Contatore`”.
4. La quarta riga dice: “Questa è la fine del ciclo Do-Loop While. Finché il valore di Contatore è inferiore a 5, continua a ripetere tutte le istruzioni racchiuse tra la riga Do e la riga Loop While; altrimenti, esci dal ciclo”.

Ogni volta che il ciclo viene eseguito, il valore di `Contatore` viene aumentato di 1. Non appena il valore di `Contatore` diviene uguale a 5, la condizione `Contatore < 5` diviene False e il ciclo Do-Loop While si interrompe.

Quante volte viene ripetuto il ciclo Do-Loop While?

A prescindere dalla condizione di un ciclo Do-Loop While, esso viene sempre eseguito almeno una volta. Se la condizione resta True, il ciclo Do-Loop While può ripetersi all’infinito (o finché vi dimenticate di pagare la bolletta della luce).

Per evitare un ciclo Do-Loop While infinito, bisogna accertarsi che almeno un’istruzione all’interno del vostro ciclo Do-Loop While modifichi il valore True o False della condizione che controlla il ciclo.

Quando utilizzare un ciclo Do-Loop While

Un ciclo Do-Loop While può essere utilizzato ogni volta che è necessario fare quanto segue.

- ✓ Ripetere un ciclo almeno una volta.
- ✓ Ripetere un ciclo finché una determinata condizione resta True.

Scegliete con molta attenzione i vostri cicli, dato che possono sembrare simili ma funzionano in modo diverso. Per evitare di fare confusione, bisogna cercare di utilizzare sempre lo stesso tipo di ciclo in tutto il programma, in modo da poterne comprendere con più facilità il funzionamento.

Mettete alla provate le vostre nuove conoscenze

1. **Quali sono le differenze principali tra un ciclo Do While e un ciclo Do-Loop While?**

   a. Le parole sono ordinate in modo diverso e utilizzano un numero diverso di consonanti.

   b. Il ciclo Do While viene eseguito zero o più volte e si ripete soltanto se la condizione è True. A prescindere dalla condizione impostata, il ciclo Do-Loop While viene eseguito almeno una volta e, se la condizione rimane True, può essere ripetuto all'infinito.

   c. Entrambi i cicli vengono eseguiti all'infinito, sempre più rapidamente, fino a quando il computer non si schianta contro il muro a causa dell'eccessiva velocità.

   d. Quattro dentisti su cinque raccomandano il ciclo Do While, gomma da masticare senza zucchero e dentifricio Crest.

2. **Perché è possibile che un ciclo Do While venga eseguito zero o più volte?**

   a. Perché è il numero utilizzato da un ciclo per due programmatori su tre impiegati presso Microsoft, Borland e Symantec.

   b. Nessuno lo sa, tuttavia ricordiamo di avere visto una puntata di "Misteri" in cui si chiedeva ai telespettatori di telefonare nel caso avessero informazioni utili per risolvere la questione.

   c. Perché controlla la condizione prima di venire eseguito anche una volta sola.

   d. Perché utilizza gli steroidi; in questo modo non solo viene eseguito una volta, ma lo fa anche più velocemente di qualsiasi altro ciclo che non utilizzi il doping.

Capitolo 24

I cicli Do Until e Do-Loop Until

In questo capitolo

- Utilizzare il ciclo Do Until
- Determinare quante volte viene ripetuto il ciclo Do Until
- Esaminare il funzionamento del ciclo Do-Loop Until e scoprire quando utilizzarlo

I cicli Do Until vengono ripetuti fino a quando una determinata condizione diventa True, come: "Continua a rubare dal registratore di cassa finché nessuno ti vede". Un ciclo Do Until ha questo aspetto:

```
Do Until Condizione
  Istruzioni
Loop
```

La condizione deve essere una variabile oppure un'espressione che rappresenta un valore True o False. Un ciclo Do Until può contenere una o più istruzioni.

Come funziona il ciclo Do Until

La prima volta che Visual Basic vede un ciclo Do Until, dice: "Vediamo se il valore della condizione è True o False; se è False ignoro tutte le istruzioni all'interno del ciclo Do Until, se è True eseguo le istruzioni all'interno del ciclo Do Until".

Vediamo un esempio di ciclo Do Until in grado di contare:

```
Contatore = 0
Do Until Contatore > 4
  Contatore = Contatore + 1
Loop
```

Ecco che cosa significa questo codice.

1. La prima riga dice: "Attribuisci il valore `0` a una variabile denominata `Contatore`".

2. La seconda riga dice: "Questo è l'inizio di un ciclo Do Until. Finché il valore di `Contatore > 4` è False, continua a ripetere tutte le istruzioni racchiuse tra la riga `Do` e la riga `Loop`. Altrimenti, se il valore di `Contatore > 4` è True (quando Contatore è 5), esci dal ciclo".
3. La terza riga dice: "Prendi il valore di `Contatore` e aggiungi 1. Ora memorizza questo valore nella variabile `Contatore`".
4. La quarta riga dice: "Questa è la fine del ciclo Do Until ".

Ogni volta che il ciclo viene eseguito, il valore di `Contatore` viene aumentato di 1. Non appena il valore di `Contatore` diviene uguale a 5, la condizione `Contatore > 4` diviene True e il ciclo Do Until si interrompe.

Quante volte viene ripetuto un ciclo Do Until

Il ciclo Do Until può essere ripetuto da zero a un'infinità di volte; questo ciclo continua a essere ripetuto fino a quando la sua condizione diviene True.

Si noti che questo è l'opposto dei cicli Do While e Do-Loop While; questi due cicli continuano a funzionare finché la condizione è True, mentre il ciclo Do Until funziona fino a quando la condizione diventa True (è come dire che il ciclo Do Until viene ripetuto finché la condizione è False).

Per evitare un ciclo Do Until infinito, accertatevi che almeno un'istruzione all'interno del vostro ciclo Do Until modifichi il valore True o False della condizione che controlla il ciclo.

Quando utilizzare un ciclo Do Until

Un ciclo Do Until può essere utilizzato ogni volta che si ha bisogno di fare quanto segue.

- ✓ Ripetere un ciclo zero o più volte.
- ✓ Continuare a ripetere qualcosa fino a quando una determinata condizione diventa True.

Il ciclo Do Until funziona come i due seguenti cicli Do While:

```
Do While Not Condizione        Do While Condizione = False
  Istruzioni                     Istruzioni
Loop                           Loop
```

Come funziona il ciclo Do-Loop Until

I cicli Do-Loop Until continuano a ripetersi finché una determinata condizione diviene True.

Un ciclo Do-Loop Until ha questo aspetto:

```
Do
  Istruzioni
Loop Until Condizione
```

La condizione deve essere una variabile oppure un'espressione che rappresenta un valore True o False. Un ciclo Do-Loop Until può contenere una o più istruzioni.

La prima volta che Visual Basic vede un ciclo Do-Loop Until, dice: "Esegui tutte le istruzioni all'interno del ciclo una volta, quindi verifica se il valore della condizione è True o False; se è True fermati, se è False esegui tutte le istruzioni all'interno del ciclo Do-Loop Until ancora una volta".

Esaminiamo un tipico ciclo Do-Loop Until in grado di contare:

```
Contatore = 0
Do
  Contatore = Contatore + 1
Loop Until Contatore > 4
```

Ecco che cosa significa questo codice.

1. La prima riga dice: "Crea una variabile denominata `Contatore` e imposta a `0` il suo valore".
2. La seconda riga dice: "Questo è l'inizio di un ciclo Do-Loop Until".
3. La terza riga dice: "Prendi il valore di `Contatore` e aggiungi 1. Ora memorizza questo valore nella variabile `Contatore`".
4. La quarta riga dice: "Questa è la fine del ciclo Do-Loop Until. Finché il valore di `Contatore` è 4 o meno, continua a ripetere tutte le istruzioni racchiuse tra la riga Do e la riga Loop Until; altrimenti, esci dal ciclo".

Ogni volta che il ciclo viene eseguito, il valore di `Contatore` viene aumentato di 1. Non appena il valore di `Contatore` diviene uguale a 5, la condizione `Contatore < 4` diviene False e il ciclo Do-Loop Until si interrompe.

Quante volte viene ripetuto il ciclo Do-Loop Until?

A prescindere dalla condizione di un ciclo Do-Loop Until, esso viene sempre eseguito almeno una volta. Questo ciclo continua a ripetersi fino a quando la condizione da False diventa True.

Si noti che questo è l'opposto dei cicli Do While e Do-Loop While; questi ultimi continuano a funzionare finché la condizione è True, mentre il ciclo Do-Loop Until funziona fino a quando la condizione è False.

Per evitare un ciclo Do-Loop Until infinito, bisogna accertarsi che almeno un'istruzione all'interno del ciclo Do-Loop Until modifichi il valore True o False della condizione che controlla il ciclo.

Quando utilizzare un ciclo Do-Loop Until

Utilizzate un ciclo Do-Loop Until ogni volta che avete bisogno di fare quanto segue.

- ✓ Ripetere un ciclo almeno una volta.
- ✓ Ripetere un ciclo finché una determinata condizione diventa True.

Il ciclo Do-Loop Until funziona come i due seguenti cicli Do-Loop While:

```
Do                                Do
  Istruzioni                        Istruzioni
Loop While Not Condizione         Loop While Condizione = False
```

Quando si creano dei cicli, bisogna sempre controllare che prima o poi giungano al termine e che facciano esattamente ciò che si desidera. Se qualcosa non funziona nel programma, è necessario verificare sempre che il problema non dipenda da un ciclo.

Mettete alla prova le vostre nuove conoscenze

1. **Per eseguire un ciclo finché una determinata condizione diventa True, che tipo di ciclo utilizzereste?**

   a. Un ciclo Do Until oppure un ciclo Do While Not.

   b. Un ciclo con la forma di un biscottino.

   c. Un cerchio della morte.

   d. Avete notato che se fissate la parola ciclo per un po', alla fine comincia a sembrare divertente?

2. **Quando utilizzate un ciclo Do Until e un ciclo Do-Loop Until?**

   a. Ogni volta che ho bisogno di confondermi le idee su come funzionano i diversi tipi di ciclo.

   b. Quando mi sbaglio e utilizzo il tipo di ciclo errato.

   c. Utilizzo il ciclo Do Until se devo ripetere un ciclo per zero o più volte e il ciclo Do-Loop Until se devo ripetere il ciclo almeno una volta.

   d. Quando non riesco a ricordarmi come utilizzare un ciclo Do While. Non potreste spiegarmelo ancora?

Capitolo 25

I cicli For Next che sono in grado di contare

In questo capitolo

- Utilizzare il ciclo For Next
- Contare in avanti e all'indietro
- Utilizzare l'incremento Step

Se si desidera ripetere un ciclo fino a quando una determinata condizione diventa True o False, è possibile utilizzare uno dei seguenti principi.

	Finché la condizione è = True	*Finché la condizione è = False*
Ripetizione almeno una volta	Do Loop While condizione	Do Loop Until condizione
Ripetizione da zero a più volte	Do While condizione	Do Until condizione

Tutti e quattro questi tipi di ciclo continuano a essere ripetuti fino a quando una determinata condizione diventa True o False. Se si sa già quante volte ripetere un ciclo, è possibile utilizzare il ciclo For Next.

Un ciclo For Next ha il seguente aspetto:

```
For Contatore = Inizio To Fine Step x
  Istruzioni
Next Contatore
```

Il contatore è una variabile che rappresenta un intero; `Inizio` rappresenta il primo numero assegnato al valore del contatore, `Fine` è l'ultimo numero assegnato al valore del contatore, `Step` è l'incremento; se si omette `Step` viene utilizzato come incremento il valore di default pari a 1.

Come funziona il ciclo For Next

Se si desidera ripetere il ciclo esattamente tre volte, è possibile utilizzare il seguente codice:

```
For X = 1 To 3
  Istruzioni
Next X
```

Ecco che cosa significa questo codice.

1. La prima riga dice: "Crea una variabile denominata X, imposta il suo valore a 1 e continua a ripetere il ciclo finché il valore di X diventa 1, 2 o 3. Quando il valore di X è diverso da questi valori, interrompi il ciclo."
2. La seconda riga è il punto in cui potete inserire una o più istruzioni, tra cui anche altri cicli For Next, se volete.
3. La terza riga dice: "OK, passa al prossimo valore di X (aggiungi 1 al valore di X) e torna alla prima riga; a questo punto, il valore di X è 2".

La riga seguente dice a Visual Basic di ripetere il ciclo tre volte.

```
For X = 1 To 3
```

Per default, Visual Basic conta con incrementi di 1. Tanto per divertirvi un po', potete utilizzare qualsiasi combinazione di numeri desiderata, come:

```
For X = 1209 To 1211
  Istruzioni
Next X
```

Questo ciclo For Next viene ripetuto sempre tre volte, sebbene ciò non sia di immediata evidenza:

```
La prima volta, X = 1209.
La seconda volta, X = 1210.
La terza e ultima volta, X = 1211.
```

È possibile utilizzare anche numeri particolari, se hanno senso all'interno del vostro programma. Per esempio, potreste contare il numero degli impiegati:

```
For NumeroImpiegati = 11250 To 11290
   ' Utilizzare il valore di NumeroImpiegati per eseguire
   ' una ricerca in un database degli impiegati e stampare
   ' il loro curriculum
Next NumeroImpiegati
```

In questo caso, le istruzioni all'interno del ciclo For Next utilizzano il valore di `NumeroImpiegati` per trovare un impiegato specifico.

Se è necessario ripetere il ciclo un numero determinato di volte, per esempio cinque, è possibile utilizzare un metodo più semplice e diretto. Vediamolo:

```
For X = 1 To 5
  Istruzioni
Next X
```

È meglio utilizzare metodi complessi soltanto se all'interno del ciclo For Next dovranno essere utilizzate cifre specifiche, come:

```
For Contatore = 3492 To 12909
  Istruzioni
Next Contatore
```

Contare in avanti e all'indietro

Solitamente il ciclo For Next conta in avanti con incrementi di 1. Per contare utilizzando un incremento diverso, dovete specificarne il valore. Per esempio:

```
For contatore = inizio To fine Step incremento
  Istruzioni
Next contatore
```

Aggiungendo l'istruzione `Step` si dice a Visual Basic: "Invece di contare in avanti con incrementi di 1, conta utilizzando il valore specificato come incremento indicato dopo la parola Step".

Se si vogliono utilizzare incrementi di 16, ecco il codice:

```
For X = 0 To 32 Step 16
  Istruzioni
Next X
```

Questo ciclo For Next viene ripetuto solo tre volte.

La prima volta, `X = 0`

La seconda volta, `X = 16`

La terza e ultima volta, `X = 32`

È possibile anche contare all'indietro. Per esempio, per contare all'indietro tre volte, potete utilizzare il seguente codice:

```
For X = 3 To 1 Step -1
  Istruzioni
Next X
```

Ecco che cosa significa questo codice.

1. La prima riga dice: "Crea una variabile denominata X, imposta il suo valore a 3 e conta all'indietro per -1".
2. La seconda riga contiene una o più istruzioni da seguire.
3. La terza riga dice: "Scegli il valore successivo di X. Dato che si sta contando all'indietro con incrementi di -1, il valore successivo di X sarà X-1. La seconda volta, X sarà 2".

Sebbene Visual Basic non si formalizzi sul metodo adottato, è bene scegliere sempre il metodo più semplice possibile; in questo modo si riuscirà sempre a capire velocemente quante volte verrà ripetuto il ciclo For Next.

Contate all'indietro o utilizzate numeri diversi da 1 (incrementi di 3, 5, 16 e così via), solo se ciò è richiesto dalle istruzioni presenti nel ciclo For Next. In caso contrario, non farete altro che rendere il programma più difficile da leggere.

Vediamo, quindi, che cosa succede se si scrive un ciclo For Next analogo al seguente:

```
For J = 1 To 7 Step 5
  Istruzioni
Next J
```

Ecco che cosa significa questo codice.

1. Questo ciclo For Next viene ripetuto due volte; la prima volta il valore di J è 1.
2. La seconda volta il valore di J è 1 + 5 (dato che il valore di Step è 5), ovvero 6.
3. Prima di essere ripetuto per la terza volta, il ciclo modifica il valore di J a 6 + 5, ovvero 11. Dato che 11 è maggiore dell'intervallo specificato per J (1 To 7), il ciclo For Next si rifiuta di ripetere le istruzioni per la terza volta ed esce.

Accorgimenti per l'utilizzo del ciclo For Next con l'incremento Step

Per poter contare, un ciclo For Next deve creare la propria variabile. Per esempio, il seguente codice crea una variabile denominata XYZ che conta con incrementi di 10:

```
For XYZ = 1 To 50 Step 10
  Istruzioni
Next XYZ
```

Quest'altro codice, invece, crea una variabile denominata TUV che conta utilizzando incrementi di 1,5:

```
For TUV = 1 To 7 Step 1,5
  Istruzioni
Next TUV
```

Solitamente il ciclo For Next utilizza numeri interi, come 1, 2, 5 o 58, per cui non è difficile determinare quanti cicli verranno eseguiti. Il primo codice d'esempio conta utilizzando incrementi di 10 e, di conseguenza, il numero delle ripetizioni è piuttosto semplice da determinare (il ciclo viene ripetuto cinque volte).

Il secondo esempio, invece, utilizza incrementi di 1,5; dato che si tratta di un incremento decimale, è più difficile stabilire quante volte verrà ripetuto il ciclo (cinque volte). Quando si utilizza l'incremento Step, è bene servirsi di numeri interi in modo da poter capire con maggiore facilità quante volte verrà ripetuto un ciclo.

Quando utilizzate un ciclo For Next, non (lo ripetiamo, non) modificate mai il valore della variabile all'interno del ciclo. In caso contrario, il ciclo si ingarbuglierà, come viene illustrato nel seguente esempio:

```
For X = 1 To 5
  X = 3
Next X
```

Ecco che cosa succede.

1. La prima riga dice: "Crea una variabile denominata X e imposta il suo valore a 1".
2. La seconda riga dice: "Attribuisci il valore 3 a una variabile denominata X".
3. La terza riga dice: "Somma 1 al valore di X. Dato che X è uguale a 3, rendi X uguale a 4".

Dato che alla fine di ogni ciclo X è sempre uguale a 4, questo ciclo For Next si trasforma in un ciclo infinito, che non si fermerà mai. Quando utilizzate il ciclo For Next, controllate sempre che nessuna istruzione all'interno del ciclo modifichi la variabile di conteggio; in caso contrario, ve ne pentirete...

Quando utilizzare un ciclo For Next

Utilizzate un ciclo For Next ogni volta che volete eseguire un ciclo per un determinato numero di volte.

Naturalmente è possibile utilizzare anche altri tipi di ciclo per contare; i due cicli seguenti vengono ripetuti esattamente sei volte:

```
X = 0                    For X = 1 To 6
Do While X < 6                   Istruzioni
  X = X + 1              Next X
  Istruzioni
Loop
```

Mettete alla prova le vostre nuove conoscenze

1. Quante volte verrà ripetuto il seguente ciclo For Next?

```
For ID = 15 To 1 Step -1

     Istruzioni

Next ID
```

a. Quindici volte.

b. Una volta, ma quindici volte più rapidamente.

c. Zero o più volte o qualcosa del genere. Aspetta un attimo. Forse sono rimasto nella lezione sbagliata.

d. Nessuna, perché solo i ricorsi storici possono ripetersi.

2. Qual è il vantaggio principale di un ciclo For Next rispetto a un ciclo Do While?

a. Un ciclo Do While è più difficile da utilizzare e un ciclo For Next non funziona proprio.

b. Dipende tutto dai punti di vista. Tutte le cose sono belle se ci amiamo a vicenda e viviamo in pace e in armonia.

c. È possibile specificare quante volte dovrà essere ripetuto un ciclo.

d. Non esiste alcun vantaggio nell'imparare Visual Basic. Fareste meglio a studiare C++ o Java.

Notate, però, quanto il ciclo For Next sia semplice e ordinato rispetto al ciclo Do While. Ci sono un'infinità di sistemi per scrivere un programma che funziona (e un numero ancora superiore di modi per scrivere un programma che non funziona), ma il metodo più semplice è di solito il migliore.

Provate anche voi

Se non avete voglia di digitare il codice, studiate quello relativo a questo capitolo memorizzato sul CD-ROM accluso.

Questo semplice programma esegue un ciclo For Next e visualizza il valore di X sullo schermo a ogni ripetizione. Per creare questo programma da soli, utilizzate le impostazioni definite nella tabella di seguito riportata.

Oggetto	*Proprietà*	*Valore*
Form	Caption	Un esempio di ciclo For-Next
Comando1	Caption	Successione di X
	Name	cmdSuccessione

Digitate quanto segue nella finestra Codice:

```
Private Sub cmdSuccessione_Click()
  For X = 1 To 10 Step 2
     Print X
  Next X
End Sub
```

Questo programma non fa nulla di più eccitante che ripetersi cinque volte e visualizzare ogni volta il valore di X sulla form.

Capitolo 26

Cicli nidificati e uscite rapide

In questo capitolo

- Utilizzare i cicli nidificati
- Far funzionare i cicli nidificati
- Uscire velocemente dai cicli

Per ottenere il massimo della flessibilità e della complessità, potete inserire un ciclo nell'altro per creare una serie infinita di cicli. Ogni volta che racchiudete un ciclo nell'altro si parla di *cicli nidificati*. Quindi, quale ciclo verrà eseguito e completato per primo? La risposta è semplice.

Utilizzare i cicli nidificati

Nei cicli nidificati, quello interno si conclude prima, come mostra l'esempio:

```
Do While Impiegato = "Supervisore"
  For J = 1 To 5
     Istruzioni
  Next J
Loop
```

Ecco che cosa significa questo codice.

1. La prima riga dice: "Controlla che il valore della variabile `Impiegato` sia uguale a `Supervisore`; in caso affermativo, passa alla seconda riga, altrimenti ignora del tutto il ciclo For Next e passa semplicemente alla quinta riga".
2. La seconda riga dice: "Crea una variabile denominata J e imposta il suo valore a 1".
3. La terza riga dice: "Segui queste istruzioni, qualsiasi esse siano".
4. La quarta riga dice: "Aumenta il valore di J di 1 e ritorna alla prima riga. Continua a farlo finché il valore di J non diventa maggiore di 5".
5. La quinta riga dice: "Questa è la fine del ciclo Do While. Torna alla prima riga e ricontrolla il valore della variabile `Impiegato`".

Nell'esempio precedente, il ciclo For Next termina prima del ciclo Do While. Il ciclo For Next viene eseguito completamente una volta ad ogni esecuzione del ciclo Do While.

Far funzionare i cicli nidificati

Naturalmente Visual Basic vi concede carta bianca e vi permette di raggruppare tutti i cicli che volete. Quando si creano dei cicli nidificati, è consigliabile applicare dei rientri a ciascun ciclo, in modo da poter individuare più facilmente dove comincia e dove finisce ogni ciclo. Per esempio, si osservi l'aspetto confuso dei cicli nidificati senza rientri:

```
Do While Nome = "Renzo"
Do
For K = 20 To 50 Step 10
Do
Do Until Sesso = "Maschile"
' Modifica qualche variabile
Loop
Loop While Età > 21
Next K
Loop Until Cognome = "Storti"
Loop
```

Ecco l'aspetto dello stesso codice con i rientri:

```
Do While Nome = "Renzo"
  Do
     For K = 20 To 50 Step 10
          Do
               Do Until Sesso = "Maschile"
               ' Modifica qualche variabile
               Loop
          Loop While Età > 21
     Next K
  Loop Until Cognome = "Storti"
Loop
```

Dal punto di vista del computer, entrambi gli esempi funzionano allo stesso modo, ma dal punto di vista del programmatore i cicli nidificati che utilizzano i rientri sono molto più semplici da leggere e da capire.

Con un numero così elevato di cicli nidificati, verificate che i cicli interni non confondano inavvertitamente le condizioni o le variabili di calcolo dei cicli più esterni. In caso contrario, si potrebbe generare un ciclo infinito ed essere costretti a esaminare tutti i cicli.

Un altro problema che può impedire ai cicli nidificati di funzionare correttamente è rappresentato da eventuali grovigli, come nell'esempio seguente:

```
For K = 1 To 4
  For J = 2 To 20 Step 2
Next K
  Next J
```

In questo esempio due cicli For Next si intrecciano, dato che il primo ciclo termina prima che possa concludersi il secondo, ovvero quello più interno. Per fortuna Visual Basic riuscirà a individuare questo errore prima che possiate eseguire il codice BASIC, per cui sarà semplice eseguire la correzione.

Uscite rapide dai cicli

Un ciclo Do continua a ripetersi fino a quando una determinata condizione diventa True o False. Un ciclo For Next viene eseguito fino al termine del conteggio. Ma che cosa si può fare nel caso si debba uscire da un ciclo prima del tempo? Provate a utilizzare il comando magico Exit.

Per abbandonare un ciclo Do, utilizzate il comando Exit Do, come illustrato nel seguente esempio:

```
X = 0
Do While X < 6
  X = X +1
  If X = 4 Then Exit Do
Loop
```

Questo ciclo Do While continua a essere ripetuto finché il valore della variabile X è inferiore a 6. Nel momento stesso in cui il valore di X diventa uguale a 4, Visual Basic esegue il comando Exit Do.

Il comando Exit Do termina il ciclo anche se il valore di X è ancora inferiore a 6.

Per uscire da un ciclo For Next, si utilizzi questa istruzione:

```
Exit For
```

Per esempio:

```
For Y = 1 To 100
If Y = 50 Then Exit For
Next Y
```

Solitamente questo ciclo For Next viene ripetuto 100 volte, ma la seconda riga dice a Visual Basic di uscire dal ciclo For Next non appena il valore di Y è uguale a 50, anche se il valore di Y è ancora inferiore a 100.

Di solito (notare che *di solito* è stato evidenziato), sarebbe meglio offrire un sistema per abbandonare prima del tempo un ciclo, nel caso l'utente desideri fare qualcos'altro. Tuttavia, accertatevi che l'utilizzo di Exit Do ed Exit For non interrompa il ciclo prima di quanto desideriate; in caso contrario, avrete creato un altro errore da rintracciare ed eliminare.

Se si utilizzano i comandi Exit Do/For all'interno di un ciclo nidificato in un altro, questi comandi garantiranno l'uscita solo dal ciclo attivo e poi riporteranno il controllo al ciclo più esterno.

Mettete alla prova le vostre nuove conoscenze

1. Qual è il limite al numero di cicli che potete nidificare?

a. Teoricamente, il numero è infinito. In pratica, il numero è rappresentato dalla vostra disponibilità a digitare, sebbene più è elevato il numero dei cicli nidificati, più difficile diventa capire che cosa fa ciascuno di essi.

b. Il numero di cicli è limitato dal vostro budget annuale, come specificato da Microsoft quando avete spedito la scheda di registrazione.

c. Cinque.

d. Scoperto da Einstein, il limite dei cicli nidificati è uguale al valore che misura la velocità della luce.

2. Che cosa è necessario fare per rendere i cicli nidificati più semplici da leggere e da capire?

a. Evitare di utilizzare i cicli, le strutture di controllo, le variabili e tutto ciò che richiede un ragionamento.

b. Assolutamente nulla. Se gli altri non riescono a capire i miei cicli nidificati, peggio per loro.

c. Evitare i rientri, dato che soltanto i dilettanti si affidano a trucchetti simili quando scrivono i programmi.

d. Utilizzare tantissimi rientri per rendere chiaramente identificabili l'inizio e la fine di ciascun ciclo.

Parte VII

Scrivere sottoprogrammi

In questa parte...

È giunto il momento di imparare come suddividere un programma Visual Basic in programmi più piccoli, che risultino più semplici da scrivere, modificare e capire. Invece di scrivere un unico e gigantesco programma monolitico (che è come scolpire un palazzo da un singolo pezzo di granito), i sottoprogrammi vi permettono di creare programmi piccoli e di incollarli insieme per creare un programma più grande (come utilizzare i mattoni per costruire una casa).

Dividendo un programma di ampie dimensioni in diverse parti più piccole, potete controllarle una per una. Se riuscirete a padroneggiare le operazioni svolte dal vostro programma una dopo l'altra, non rischierete di diventare pazzi cercando di creare un programma colossale tutto d'un fiato.

Capitolo 27

Procedure generali (per non impazzire all'istante)

In questo capitolo

- Creare delle procedure generali
- Attribuire i nomi alle procedure generali
- Utilizzo delle procedure generali

Le procedure sono piccoli programmi che, uniti, danno vita a un unico programma più grande, un po' come i mattoni che compongono un muro. Visual Basic offre due tipi di procedure: le procedure di gestione degli eventi e le procedure generali.

Una *procedura di gestione degli eventi* fa parte di un oggetto dell'interfaccia utente, come un pulsante di comando, una casella di controllo o una barra di scorrimento. Le procedure di gestione degli eventi vengono eseguite solo quando un determinato evento interessa un oggetto specifico, come un clic del mouse su un pulsante di comando o una casella di controllo.

Una *procedura generale* non è legata ad alcun oggetto particolare dell'interfaccia utente e non fa nulla se prima una procedura di gestione degli eventi (o un'altra procedura generale) non le dice in modo specifico di mettersi al lavoro.

Sono necessarie le procedure di gestione degli eventi? Sì, poiché consentono alla vostra interfaccia utente di reagire. Sono necessarie le procedure generali? No; infatti esistono esclusivamente per comodità dei programmatori.

Se due o più procedure di gestione degli eventi contengono istruzioni quasi identiche, sarebbe noioso digitare gli stessi dati più volte; inoltre, nel caso dovesse capitare di dover modificare tali istruzioni, si sarebbe costretti a intervenire su tutte le procedure di gestione degli eventi che utilizzano quelle stesse istruzioni.

Come alternativa è possibile utilizzare le procedure generali. Lo scopo di queste procedure è proprio quello di conservare in un unico posto le istruzioni utilizzate

di frequente. In questo modo, se fosse necessario modificare le istruzioni, basterebbe intervenire in un unico punto.

Creare una procedura generale

Dopo che avrete disegnato tutti gli oggetti che compongono la vostra interfaccia utente, Visual Basic vi permetterà di creare in modo automatico procedure di gestione degli eventi vuote per tutti gli oggetti presenti sull'interfaccia.

Purtroppo, Visual Basic non offre un metodo altrettanto veloce per creare una procedura generale e quindi dovrete generarla da soli. È possibile creare e salvare le procedure generali in due tipi di file.

- ✓ File FRM (form).
- ✓ File BAS (modulo).

Quando salvate una procedura generale in un file FRM (form), la procedura generale può essere utilizzata soltanto dalle procedure di gestione degli eventi oppure dalle procedure generali memorizzate nello stesso file FRM (l'unica eccezione è che qualunque parte del vostro programma può utilizzare una procedura generale memorizzata in un file FRM, ma solo se quel file è caricato in memoria in quel momento). Quando invece si salva una procedura generale in un file BAS (modulo), la stessa potrà essere utilizzata da tutte le procedure di gestione degli eventi e da tutte le procedure generali che costituiscono il programma Visual Basic.

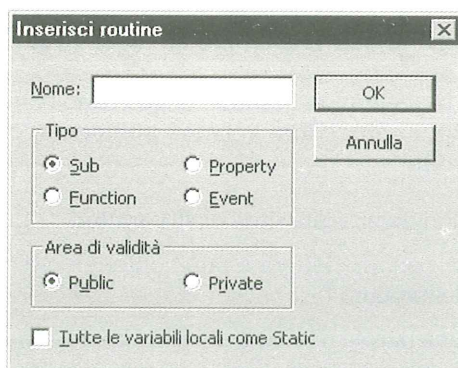
Se salvate le procedure generali in un file BAS, potete creare una libreria di procedure generali utili, pronte per essere inserite nei vari programmi Visual Basic che scriverete. Se le procedure generali devono essere utilizzate soltanto per un programma specifico, memorizzatele in un file FRM.

Per creare e salvare una procedura generale in un file FRM (form), è possibile utilizzare la seguente procedura.

1. **Fate clic su un file form nella finestra Progetto e poi aprite la finestra Codice. A questo scopo premete F7, selezionate Visualizza-Codice oppure fate clic sull'icona Visualizza codice.**
2. **Selezionate (Generale) nella casella di riepilogo Oggetto.**
3. **Selezionate Strumenti-Inserisci routine...**

   Visual Basic visualizza la finestra di dialogo Inserisci routine, riportata nella Figura 27.1.
4. **Selezionate il pulsante di opzione Sub e poi digitate il nome della procedura nella casella Nome e premete INVIO oppure fate clic su OK.**

   Visual Basic visualizza una procedura generale vuota.

Figura 27.1 La creazione di una procedura generale con la finestra di dialogo Inserisci routine

Per creare e salvare una procedura generale in un nuovo file BAS (modulo), utilizzate la procedura di seguito riportata.

1. **Selezionate Progetto-Inserisci modulo.**

   Appare la finestra di dialogo Inserisci modulo.

2. **Fate clic sull'icona Modulo e poi fate clic su Apri.**
3. **Selezionate Strumenti-Inserisci routine...**

   Visual Basic visualizza la finestra di dialogo Inserisci routine, riportata nella Figura 27.1.

4. **Controllate che sia stato selezionato il pulsante di opzione Sub e poi digitate il nome della procedura nella casella Nome e premete INVIO oppure fate clic su OK.**

   Visual Basic visualizza una procedura generale vuota.

Per creare una procedura generale e salvarla in un file BAS esistente, si utilizzi la procedura di seguito riportata.

1. **Selezionate Visualizza-Gestione progetti, premete CTRL+R oppure fate clic sull'icona Gestione progetti nella barra degli strumenti.**
2. **Fate clic sul file BAS nel quale volete salvare la procedura generale e poi fate clic sull'icona Visualizza codice.**

   Visual Basic visualizza la finestra Codice di quel particolare file BAS.

3. **Selezionate Strumenti-Inserisci routine...**

   Visual Basic visualizza la finestra di dialogo Inserisci routine.

4. **Con il pulsante di opzione Sub selezionato, digitate il nome della procedura nella casella Nome e premete INVIO oppure fate clic su OK.**

   Visual Basic visualizza una procedura generale vuota.

Attribuire i nomi alle procedure generali

Contrariamente ai nomi delle procedure di gestione degli eventi che identificano il nome dell'oggetto e l'evento, le procedure generali possono avere un nome qualsiasi, ma con i seguenti limiti.

- ✓ Il nome deve avere una lunghezza massima di 40 caratteri.
- ✓ Il nome deve cominciare con una lettera e può contenere solo lettere, numeri e il carattere di sottolineatura (_).
- ✓ Il nome non può essere una parola riservata di Visual Basic come End, Sub o Private.

In linea di principio sarebbe meglio utilizzare nomi che descrivano la funzione della procedura generale, per esempio:

```
RadiceQuadrata
Chiedi4Password
VisualizzaFinestra
```

Nella finestra Codice questi nomi di procedura verrebbero visualizzati nel seguente modo:

```
Public Sub RadiceQuadrata()
End Sub
```

e

```
Public Sub Chiedi4Password()
End Sub
```

e

```
Public Sub VisualizzaFinestra()
End Sub
```

Notate che il nome completo di una procedura generale comprende quattro parti.

- ✓ `Public` (o `Private`).
- ✓ `Sub`.
- ✓ Il nome della procedura generale.
- ✓ Una coppia di parentesi, ().

La parola `Public` comunica a Visual Basic che la procedura generale è pubblica; ciò significa che se è memorizzata in un file BAS, qualsiasi evento o procedura memorizzati in altri file FRM o BAS potranno utilizzarla (se non volete che le procedure memorizzate in altri file usino procedura generale, utilizzate la parola Private).

La parola `Sub` identifica la vostra subroutine come procedura. (Se alla parola Sub sostituite la parola Function, potrete creare una funzione generale. Vi parlerò delle funzioni nel Capitolo 29).; il nome è la denominazione che dovrete utilizzare all'interno delle procedure di gestione degli eventi e delle altre procedure generali per chiamare la procedura generale. *Chiamare* una procedura generale significa dire a una particolare procedura: "OK, fai qualcosa subito!".

La coppia di parentesi viene definita elenco degli argomenti (il tema è affrontato nel Capitolo 28). Le procedure generali più semplici hanno un elenco degli argomenti vuoto, rappresentato solo da una coppia di parentesi.

Utilizzare una procedura generale

Una procedura generale contiene una o più istruzioni. Quando un'altra procedura vuole utilizzare le istruzioni memorizzate in essa, chiama quest'ultima per nome.

Per chiamare una procedura, sono utilizzabili due sistemi. Si può specificare il nome della procedura:

```
NomeProcedura
```

Oppure si può indicare il nome della procedura insieme alla parola `Call`:

```
Call NomeProcedura
```

Se volete utilizzare una procedura generale memorizzata in un altro file FRM, potete chiamarla specificando il nome della form e poi il nome della procedura:

```
NomeForm.NomeProcedura
```

Oppure potete indicare il nome della procedura insieme alla parola `Call`:

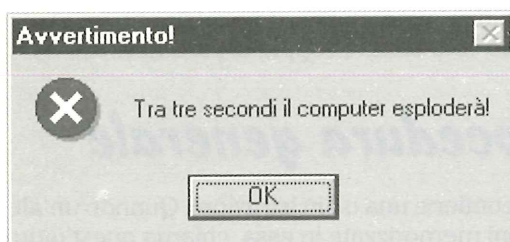
```
Call NomeForm.NomeProcedura
```

Il computer non si formalizza sul metodo utilizzato: sceglietene uno e utilizzate sempre lo stesso, in modo da rendere il programma più semplice da leggere.

Specificare solo il nome della procedura è più semplice che utilizzare anche la parola `Call`, tuttavia quest'ultima aiuta a identificare tutte le chiamate di procedure inserite nel programma. Esaminiamo, per esempio, la seguente procedura generale memorizzata in una form chiamata frmMainForm.

```
Public Sub Avvertimento()
  MsgBox "Tra tre secondi il computer esploderà!", 16,
"Avvertimento!"
End Sub
```

Questa procedura generale si limita a visualizzare una finestra di dialogo con il messaggio "Tra tre secondi il computer esploderà!", come mostra la Figura 27.2.

Figura 27.2 Una finestra di dialogo creata da una procedura generale

Se una procedura di gestione degli eventi memorizzata in una form differente volesse utilizzare o chiamare questa procedura generale, potrebbe fare così:

```
Public Sub cmdAllerta_Click()
  frmMainForm.Avvertimento
End Sub
```

Se la procedura di gestione degli eventi avesse utilizzato il metodo `Call` per chiamare la procedura, il codice sarebbe stato questo:

```
Public Sub cmdAllerta_Click()
  Call frmMainForm.Avvertimento
End Sub
```

Entrambe queste procedure di gestione degli eventi producono il seguente risultato:

```
Sub cmdAllerta-Click()
     MsgBox "Tra tre secondi il computer esploderà!", 16,
"Avvertimento!"
End Sub
```

Benché questo sia un esempio semplice, risulta evidente che inserendo due o più istruzioni in una procedura generale si evita di dover digitare queste stesse istruzioni un'infinità di volte in più punti. In questo modo, è sufficiente chiamare un'unica procedura generale.

Mettete alla prova le vostre nuove conoscenze

1. **Perché utilizzare una procedura generale?**

   a. Perché una procedura più specifica non andrebbe bene.

   b. Per memorizzare istruzioni utilizzate di frequente in un unico luogo in modo che la procedura sia semplice da modificare.

   c. Per impedire che le procedure di gestione degli eventi si sentano sole.

   d. Non esiste alcun motivo per utilizzare le procedure generali: i programmatori in gamba non hanno bisogno di simili espedienti.

2. **Quali sono i due metodi per utilizzare o chiamare una procedura generale?**

   a. Telefonare o spedire una lettera.

   b. Dichiarare il nome della procedura generale oppure inserire la parola Call prima del nome della procedura generale.

   c. Far visita al proprio veggente preferito e partecipare a una seduta spiritica.

   d. Indurre quello tra voi con la voce più potente a chiamare a voce alta la procedura al posto vostro.

Capitolo 28

Passare gli argomenti

In questo capitolo

- Inviare gli argomenti a una procedura
- Passare degli argomenti per valore
- Uscire anticipatamente da una procedura

Quando una procedura ne chiama un'altra generale, lo fa per nome. In questo modo si dice alla procedura: "Ehi, svegliati e comincia a fare qualcosa".

Spesso la procedura generale non necessita di altre istruzioni quando viene chiamato il suo nome; talvolta, però, la procedura generale chiamata ha bisogno di ulteriori informazioni prima di poter agire.

Una procedura può chiamare una procedura generale e fornirle dei dati con i quali lavorare; questi dati vengono definiti *argomento*. In sostanza, la prima procedura dice: "Smettila di sollevare obiezioni. Ecco tutte le informazioni di cui hai bisogno. Ora, datti da fare".

Perché utilizzare gli argomenti?

Un argomento è costituito dai dati (numeri, stringhe oppure variabili che rappresentano un numero o una stringa) dei quali una procedura generale ha bisogno per funzionare. Se utilizzate gli argomenti, potete scrivere una procedura singola, non specifica, in grado di sostituire due o più procedure generali specializzate.

Per esempio, potete creare due procedure generali in questo modo:

```
Public Sub VisualizzaAvvertimento()
  txtLeggimi.Text = "Allarme! Si è verificata la fusione del
nocciolo!"
End Sub
```

e

```
Public Sub VisualizzaAttenzione()
  txtLeggimi.Text = "Allarme! Spegnere subito i reattori!"
End Sub
```

Per utilizzare entrambe le procedure, dovete chiamarle adottando uno dei seguenti metodi.

- ✓ `VisualizzaAvvertimento` oppure `Call VisualizzaAvvertimento`.
- ✓ `VisualizzaAttenzione` oppure `Call VisualizzaAttenzione`.

Tuttavia creare due procedure che fanno quasi la stessa cosa è un lavoro noioso che fa perdere tempo, una tecnica perfetta quando un lavoro non vi piace.

Potete invece sostituire entrambe queste procedure con una sola, così:

```
Public Sub Visualizza(Messaggio As String)
  txtLeggimi.Text = Messaggio
End Sub
```

Questa nuova procedura dice: "Crea una variabile denominata `Messaggio` che contiene le informazioni fornite da un'altra procedura e, qualsiasi sia il valore, inseriscilo nella proprietà `Text` della casella di testo `txtLeggimi`.

Quando utilizzate una procedura generale, potete scegliere il tipo di testo richiamando la procedura Visualizza, così:

```
Visualizza ("Allarme! Si è verificata la fusione del
nocciolo")
```

oppure

```
Visualizza ("Allarme! Spegnere subito i reattori")
```

Inviare gli argomenti a una procedura

Per chiamare una procedura e inviarle un argomento, potete utilizzare uno di questi tre metodi.

- ✓ `NomeProcedura Argomento`
- ✓ `NomeProcedura (Argomento)`
- ✓ `Call NomeProcedura (Argomento)`

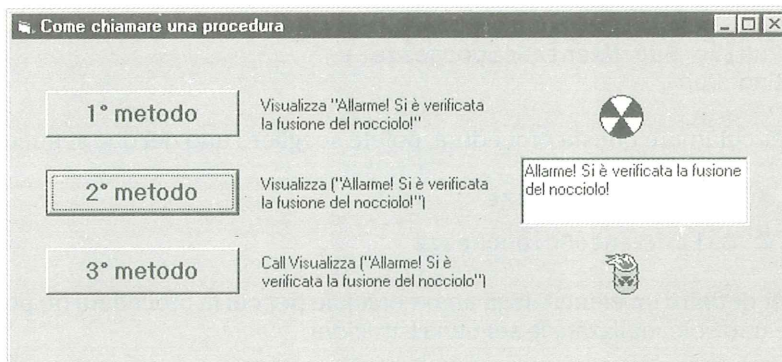
Si supponga di avere la seguente procedura generale:

```
Public Sub Visualizza(messaggio)
  txtLeggimi.Text = Messaggio
End Sub
```

Per chiamare la procedura precedente e visualizzare il messaggio `"Allarme! Si è verificata la fusione del nocciolo!"`, è possibile utilizzare uno di questi tre metodi.

- ✓ `Visualizza "Allarme! Si è verificata la fusione del nocciolo!"`
- ✓ `Visualizza ("Allarme! Si è verificata la fusione del nocciolo!")`
- ✓ `Call Visualizza ("Allarme! Si è verificata la fusione del nocciolo!")`

Tutti e tre i metodi sono equivalenti, come illustrato nella Figura 28.1.

Figura 28.1 La chiamata di una procedura generale

Vediamo ora che cosa succede chiamando la procedura generale precedente.

1. Innanzitutto dice a Visual Basic: "Trova una procedura denominata `Visualizza` e trasmettile un argomento". In questo caso l'unico argomento è la stringa: `"Allarme! Si è verificata la fusione del nocciolo!"`.
2. Visual Basic trova una procedura generale denominata `Visualizza`, che dice di assegnare l'argomento ottenuto alla variabile `Messaggio`.
3. Poi la procedura `Visualizza` dice: "Inserisci il valore di `Messaggio` nella proprietà `Text` di una casella di testo denominata `txtLeggimi`. Dato che il valore di `Messaggio` è la stringa `"Allarme! Si è verificata la fusione del nocciolo!"`, inserisci questo valore nella proprietà `Text`".

Se aveste chiamato la procedura nel seguente modo:

```
Visualizza ("Allarme! Spegnere subito i reattori!")
```

la proprietà `Text` della casella di testo `txtLeggimi` avrebbe contenuto `"Allarme! Spegnere subito i reattori!"`.

Utilizzando la stessa procedura ma con argomenti diversi sono state sostituite due procedure generali specializzate con una singola procedura generale.

Accettare gli argomenti

Prima che una procedura generale possa accettare gli argomenti, dovete definire l'elenco di questi. In sostanza l'elenco definisce il numero di argomenti che la procedura generale può accettare.

Per esempio, per definire una procedura generale che non accetti argomenti, potete utilizzare una serie di parentesi vuote, in questo modo:

```
Public Sub NienteSciocchezze()
End Sub
```

Per chiamare questa procedura, potete scegliere uno dei due seguenti metodi.

- ✓ `NienteSciocchezze`
- ✓ `Call NienteSciocchezze`

Per definire un elenco degli argomenti tale per cui la procedura ne possa accettare uno solo, utilizzate le seguenti istruzioni:

```
Public Sub Sciocchezze(Qualcosa)
End Sub
```

In questo caso, la variabile `Qualcosa` è, per default, una variabile di tipo Variant, (per informazione sui tipi di dati leggete il Capitolo 15), vale a dire che può rappresentare sia un numero che una stringa. Per chiamare questa procedura e attribuirle l'argomento 4, potete utilizzare uno di questi metodi.

- ✓ `Sciocchezze 4`
- ✓ `Sciocchezze (4)`
- ✓ `Call Sciocchezze(4)`

Per definire un elenco degli argomenti tale per cui la procedura ne accetti due o più, dovete specificare una variabile per ogni argomento e separarle con una vir-

gola. Per esempio, quanto segue definisce una procedura in grado di ricevere tre argomenti:

```
Public Sub Chiacchiere(Messaggio, Risposta, Pettegolezzi)
End Sub
```

Questo elenco definisce tre argomenti che possono rappresentare un numero o una stringa. Per chiamare questa procedura e attribuirle gli argomenti 30, “Ciao” e 12,9 avete due possibilità.

- ✓ `Chiacchiere 30, "Ciao", 12.9`
- ✓ `Call Chiacchiere(30, "Ciao", 12.9)`

Non esiste un limite specifico al numero di argomenti che una procedura può accettare. Tuttavia più lungo è il vostro elenco degli argomenti, più complessa dovrà essere la vostra procedura e maggiori saranno le probabilità che vi confondiate nel tentativo di comprendere che cosa fa esattamente la procedura.

Definire i tipi di argomento

Oltre a definire il numero di argomenti in un elenco, potete anche specificare il tipo di dati che ogni argomento può accettare.

Per esempio potete definire un argomento che accetti soltanto uno dei seguenti tipi di dati (per informazione sui tipi di dati leggete il Capitolo 15).

- ✓ Integer.
- ✓ Long.
- ✓ Single.
- ✓ Double.
- ✓ Currency.
- ✓ String.

I tipi di argomento devono essere definiti nell’elenco, per esempio:

```
Public Sub Conversione(Fahrenheit As Integer, Celsius As Integer)
```

Questa istruzione definisce due argomenti ed entrambi devono accettare un intero. Il codice di seguito riportato illustra l’unica chiamata di procedura possibile.

```
Public cmdTest_Click()
Dim X, Y As Integer
Dim A, B As String
```

```
Dim M, N As Single
  Call Conversione(X, Y)  'Questa funziona
  Call Conversione(A, B)  'A e B non sono Integer
  Call Conversione(M, N)  'M e N non sono Integer
  Call Conversione("Ciao", X)  '"Ciao" non è un Integer
End Sub
```

La ragione principale per specificare il tipo di argomento è impedire alla procedura di provare a utilizzare dati errati. Se, per esempio, la procedura si aspettasse una stringa e ottenesse un numero intero, il programma potrebbe interrompersi oppure, peggio ancora, potrebbe continuare a funzionare correttamente ma con le informazioni sbagliate. In questo caso, avreste creato un errore logico (per maggiori informazioni sugli errori logici leggete il Capitolo 20).

Sebbene non sia obbligatorio specificare il tipo di argomento, sarebbe meglio farlo sempre in modo da impedire a eventuali errori di penetrare nel vostro prezioso codice.

Problemi con l'invio di argomenti

Esistono due problemi che possono verificarsi quando si chiamano le procedure. Il primo si incontra quando il numero degli argomenti inviati non corrisponde al numero degli argomenti definiti dalla procedura. Il secondo problema si presenta invece quando i tipi di argomenti passati non corrispondono ai tipi di argomenti definiti dalla procedura.

Fornire il numero sbagliato di argomenti

Quando definite una procedura con un elenco degli argomenti, in quest'ultimo viene specificato il numero di argomenti che la procedura deve accettare. Se date il nome ma non passate il numero corretto di argomenti, la procedura non funzionerà. Per esempio:

```
Public Sub Litighiamo(Fiamme)
End Sub
```

Questa procedura si aspetta un argomento che può essere un numero oppure una stringa. Nulla di quanto segue consente alla procedura di funzionare, perché il numero degli argomenti è diverso da uno.

- `Litighiamo`
- `Litighiamo 9, "Taci!"`
- `Call Litighiamo("Perché", "Vai via!", 4500, "OK")`

Fornire il tipo errato di argomenti

In modo analogo, quando chiamate una procedura, controllate sempre che gli argomenti possiedano lo stesso tipo di dati definito nell'elenco. Per esempio:

```
Public Sub Litighiamo(Fiamme As String)
End Sub
```

Questa procedura si aspetta un argomento che deve essere di tipo String. Niente di quanto segue consente alla procedura di funzionare, dato che gli argomenti non sono di tipo String.

- ✓ `Litighiamo(78.909)`
- ✓ `Litighiamo(9)`
- ✓ `Call Litighiamo(34)`

Passare gli argomenti per valore

Per proteggere ulteriormente le procedure ed evitare ogni possibile confusione con le variabili, è possibile passare gli argomenti per valore. Solitamente, quando passate un argomento a una procedura, la nuova procedura può modificare il valore dell'argomento. È un po' come offrire a qualcuno una bibita, lasciare che ci sputi dentro e poi ce la restituisca.

Se un'altra procedura ha bisogno di determinati dati, ma non volete che i dati utilizzati da altre parti del programma siano modificati da una procedura diversa, potete passare un argomento per valore. Questo significa inviare alla procedura un argomento e concederle di modificarlo nel modo desiderato, ma limitatamente a quella particolare procedura. È come versare una parte della vostra bibita in un altro bicchiere e offrirlo a una persona diversa. Se questa persona decide di sputarci dentro, la vostra bibita resterà comunque incontaminata.

Per definire un argomento come passato per valore, utilizzate la parola chiave `ByVal` nell'elenco degli argomenti in questo modo:

```
Public Sub Vediamo(ByVal Nome As String)
```

Non è mai necessario utilizzare la parola `ByVal` per eseguire una chiamata di procedura. Supponete di avere una procedura di gestione degli eventi simile alla seguente:

```
Private Sub Comando1_Click()
Dim MiaStringa As String
  MiaStringa = "Gianni Bronco"
  Vediamo MiaStringa
  txtMessaggio.Text = MiaStringa
End Sub
```

e una procedura generale simile alla seguente:

```
Public Sub Vediamo (ByVal Nome As String)
  Nome = UCase(Nome)
  txtNuovoMessaggio.Text = Nome
End Sub
```

Ecco come Visual Basic esegue le istruzioni nella procedura di gestione degli eventi `Comando1_Click()`.

1. Quando l'utente fa clic su un pulsante di comando denominato `Comando1`, Visual Basic comincia a seguire le istruzioni presenti nella procedura di gestione degli eventi `Comando1_Click()`.
2. La seconda riga nella gestione degli eventi `Comando1_Click()` comunica a Visual Basic di dichiarare una variabile denominata `MiaStringa` che può contenere solo valori di tipo stringa.
3. La terza riga attribuisce la stringa `"Gianni Bronco"` alla variabile `MiaStringa`.
4. La quarta riga chiama la procedura generale `Vediamo` e le invia il valore della variabile `MiaStringa` come argomento.
5. Visual Basic passa immediatamente alla procedura generale denominata `Vediamo`. La parola chiave `ByVal` dice a Visual Basic che qualunque modifica apportata all'argomento ricevuto viene isolata all'interno della procedura generale `Vediamo`.
6. La prima riga della procedura generale `Vediamo` definisce una nuova variabile denominata `Nome` che contiene un valore di tipo stringa.
7. La seconda riga della procedura generale `Vediamo` dice di utilizzare la parola chiave `UCase` per trasformare in maiuscole tutte le lettere della variabile `Nome` e per memorizzare il risultato nuovamente nella variabile `Nome`. In questo caso, il valore della variabile `Nome` è ora `"GIANNI BRONCO"`.
8. La terza riga della procedura generale `Vediamo` dice di visualizzare il valore Nome nella proprietà `Text` di una casella di testo denominata `txtNuovoMessaggio`. In questo caso, la casella di testo `txtNuovoMessaggio` visualizza `"GIANNI BRONCO"`.
9. La quarta riga della procedura generale `Vediamo` dice che questa è la fine della procedura generale e che occorre ritornare alla procedura che aveva chiamato la procedura generale `Vediamo`. Questa informazione comunica a Visual Basic di passare nuovamente alla quinta riga della procedura di gestione degli eventi `Comando1_Click()`.
10. La quinta riga della procedura di gestione degli eventi `Comando1_Click()` dice di prendere il valore di `MiaStringa` e di attribuirlo alla proprietà `Text` della casella di testo `txtMessaggio` che, in questo caso, visualizza `"Gianni Bronco"`.

11. La sesta riga della procedura di gestione degli eventi `Comando1_Click()` dice a Visual Basic di smetterla di pensare.

Se eliminate la parola chiave `ByVal` dalla procedura generale `Vediamo`, la casella di testo `txtMessaggio` visualizza `"GIANNI BRONCO"` al punto 10. La parola chiave `ByVal` impedisce semplicemente alla procedura generale di ingarbugliare le variabili utilizzate da altre procedure.

Per specificare che un argomento deve essere passato come valore, bisogna inserire la parola chiave `ByVal` davanti a ciascun argomento:

```
Public Sub ScatolaNera(ByVal X As Integer, Y As Integer)
```

Nell'esempio precedente, soltanto l'argomento `X` viene passato per valore, ma la variabile `Y` no. Per specificare che anche l'argomento `Y` deve essere passato per valore, va utilizzato il seguente codice:

```
Public Sub ScatolaNera(ByVal X As Integer, ByVal Y As Integer)
```

Uscire anticipatamente da una procedura

Di solito una procedura viene eseguita fino al completamento di tutte le istruzioni. Tuttavia si potrebbe voler uscire da una procedura prima che sia terminata.

Per uscire anticipatamente da una procedura, si può utilizzare questo codice:

```
Exit Sub
```

Per esempio, potreste avere una procedura simile alla seguente:

```
Public Sub CicloInfinito()
  X = 0
  Do
      X =X +1
      If (X = 13) Then
              Exit Sub
          End if
  Loop Until X = 25
End Sub
```

Normalmente, questo ciclo verrebbe ripetuto finché il valore di X non risulta uguale a 25. In questo caso, però, il ciclo If-Then interno al ciclo Do fa sì che questa procedura venga conclusa quando X è uguale a 13.

Provate anche voi

Se desiderate esercitarvi a passare gli argomenti, prendete il CD. Qui troverete un programma di esempio che vi consentirà di digitare un testo in una casella di testo e di vedere come il passaggio di una variabile per valore (mediante la parola chiave ByVal) determini la variazione del valore della variabile solo all'interno di quella particolare procedura. Questo programma è memorizzato sotto il nome CALL.VBP.

Mettete alla prova le vostre nuove conoscenze

1. **Spiegate perché passare gli argomenti è utile per scrivere dei programmi.**

   a. Il passaggio degli argomenti equivale a uno scaricabarile. I programmatori lo fanno sempre per evitare di assumersi responsabilità quando il progetto è in ritardo.

   b. Il passaggio degli argomenti permette di scrivere una procedura generale per sostituire due o più procedure specifiche.

   c. Gli argomenti permettono di fornire informazioni errate alle procedure; in questo modo le possibilità di distruggere il vostro progetto raddoppiano.

   d. Il passaggio di argomenti equivale alla terapia delle urla. Ciascuna parte coinvolta sostiene il proprio punto di vista fino allo sfinimento; in questo modo si evita che la gente si spari addosso in ufficio.

2. **Spiegate che cosa significa il seguente elenco degli argomenti.**

```
Public Sub Confusione
(ByVal Prendi As String, X As
Integer, Z)

End Sub
```

   a. Non ho ancora studiato la lezione; inoltre devo ripassare tutto.

   b. Finalmente capisco perché i programmatori hanno quell'aspetto e si comportano in quel modo se devono trascorrere otto ore al giorno decifrando comandi incomprensibili come questo.

   c. Qualcuno non ha commentato adeguatamente il codice in modo da renderlo comprensibile.

   d. La procedura attende tre argomenti. Il primo si chiama Prendi, viene passato per valore e deve essere del tipo String; il secondo si chiama X e deve essere del tipo Integer; il terzo si chiama Z e può contenere dati di qualsiasi tipo.

Capitolo 29

Funzioni, un tipo di sottoprogramma unico

In questo capitolo

- Chiamare le funzioni
- Definire i tipi di argomento
- Uscire anticipatamente da una funzione

Mentre le procedure non restituiscono valori e si limitano a variare gli argomenti che vengono loro passati, le funzioni restituiscono un singolo valore, che può essere assegnato a una variabile o utilizzato all'interno di una espressione. Quando avete bisogno di calcolare un unico valore, utilizzate una funzione. Quando invece volete calcolare zero o più di due valori, utilizzate una procedura.

Una funzione tipica ha il seguente aspetto:

```
Public Function NomeFunzione(ElencoArgomenti)As DataType
  NomeFunzione = QualcheValore
End Function
```

La parola `Public` dice a Visual Basic che se la funzione è memorizzata in un file BAS potrà essere utilizzata da tutte le procedure di gestione degli eventi presenti nel programma Visual Basic, mentre se è memorizzata in un file FRM potrà essere utilizzata solo dalle procedure di gestione degli eventi memorizzate in quello stesso file FRM.

La parola `Function` definisce il sottoprogramma come una funzione. Il `NomeFunzione` può essere rappresentato da qualsiasi nome valido Visual Basic, preferibilmente uno che descriva che cosa fa la funzione. L'elenco degli argomenti può contenere zero o più argomenti. DataType definisce il tipo di dati restituito dalla funzione, come Integer o String.

Visual Basic comprende numerose funzioni incorporate che potete utilizzare nei vostri programmi. La Tabella 29.1 ne elenca alcune.

Tabella 29.1 ***Alcune funzioni Visual Basic comunemente utilizzate***

Funzioni Visual Basic incorporate	*Cosa fanno*
Abs (numero)	Restituisce il valore assoluto di un numero.
Date	Restituisce la data corrente del sistema.
LCase (stringa)	Converte una stringa in caratteri minuscoli.
Sqr (numero)	Restituisce la radice quadrata di un numero.

Come creare una funzione

È possibile creare e salvare le funzioni in due tipi di file:

- ✓ file FRM (form)
- ✓ file BAS (modulo)

Quando una funzione viene salvata in un file FRM (form), detta funzione potrà essere utilizzata solo da procedure o funzioni contenute nello stesso file FRM (l'unica eccezione si verifica nel caso una parte del vostro programma possa utilizzare una funzione memorizzata in un file form FRM, ma solo se in quel momento detto file form è caricato in memoria). Quando una funzione viene salvata in un file BAS (modulo), detta funzione potrà essere utilizzata da tutte le procedure o funzioni che compongono il vostro programma in Visual Basic.

Quando salvate le funzioni in un file BAS, potete creare una libreria di utili funzioni che potrete impiegare anche in altri programmi in Visual Basic che scriverete. Se le funzioni servono solo per un particolare programma, memorizzatele in un file FRM.

Per creare e salvare una funzione in un file FRM (form), procedete come indicato qui di seguito.

1. **Fate clic su un file form nella finestra Progetto, quindi premete F7, selezionate Visualizza-Codice oppure fate clic sull'icona Visualizza codice in modo da aprire la finestra Codice.**
2. **Selezionate (Generale) nella casella di riepilogo Oggetto.**
3. **Selezionate Strumenti-Inserisci routine.**

   Visual Basic visualizza la finestra di dialogo Inserisci routine.
4. **Fate clic sul pulsante di opzione Funzione, quindi, nella casella Nome, digitate il nome della funzione e premete INVIO oppure fate clic su OK.**

   Visual Basic visualizza una funzione vuota.

Per creare e salvare una funzione in un nuovo file BAS (modulo), procedete nella maniera seguente.

1. **Selezionate Progetto-Inserisci modulo.**

   Viene così visualizzata la finestra di dialogo Inserisci modulo.

2. **Fate clic sull'icona Modulo e fate clic su Apri.**

3. **Selezionate Strumenti-Inserisci routine.**

   Visual Basic visualizza la finestra di dialogo Inserisci routine (Figura 27.1).

4. **Fate clic sul pulsante di opzione Funzione, quindi, nella casella Nome, digitate il nome della funzione e premete Invio oppure fate clic su OK.**

   Visual Basic visualizza una funzione vuota.

Per creare una funzione e salvarla in un file BAS esistente, procedete come indicato di seguito.

1. **Selezionate Visualizza-Gestione progetti, premete Ctrl+R oppure fate clic sull'icona Progetto sulla barra degli strumenti.**

2. **Fate clic sul file modulo in cui intendete salvare la procedura generale. Quindi fate clic sull'icona Visualizza Codice.**

   Visual Basic visualizza la finestra Codice relativa a quel particolare file modulo BAS.

3. **Selezionate Strumenti-Inserisci routine.**

   Visual Basic visualizza la finestra di dialogo Inserisci routine (Figura 27.1).

4. **Fate clic sul pulsante di opzione Funzione, quindi, nella casella Nome, digitate il nome della funzione e premete Invio oppure fate clic su OK.**

   Visual Basic visualizza una funzione vuota.

Determinare il valore restituito della funzione

In qualche punto, all'interno della funzione, dovrete comunicare a Visual Basic quale sia il valore che la funzione deve restituire. Tale valore, che può essere una costante o, come nell'esempio seguente, il risultato di una espressione, deve essere assegnato ad una variabile il cui nome è identico a quello della funzione.

```
Public Function IardeAMetri (Iarde As Single) As Single
Const Conversione = 0.9
  IardeAMetri = Iarde * Conversione
End Function
```

Se non si assegna alcun valore alla funzione, questa non restituirà nulla (e lo scopo di una funzione è proprio quello di restituire un valore).

Potete anche definire il tipo di dati del valore che dovrà essere restituito, come Integer, String o Currency. Per tutte le informazioni possibili sui tipi di dati, leggete il Capitolo 15.

Le tre differenze principali tra una funzione e una procedura sono le seguenti.

- ✓ Una funzione restituisce solo un valore, mentre una procedura si limita a modificare il valore degli argomenti.
- ✓ In qualche punto all'interno della funzione è necessario assegnare un valore al nome della funzione, mentre questo non avviene con una procedura.
- ✓ È possibile definire il tipo di dati che una funzione restituisce, mentre non è possibile definire il tipo di dati rappresentato da una procedura (ma è possibile definire i tipi di dati dell'elenco degli argomenti di una procedura).

Chiamare le funzioni

Chiamare una funzione è diverso dal chiamare una procedura; dato che le funzioni restituiscono un valore, questo dovrà essere assegnato ad una variabile:

```
Public Function IardeAMetri (Iarde As Single)
Const Conversione = 0.9
  IardeAMetri = Iarde * Conversione
End Function
Private Sub cmdConversione_Click()
Dim Metri As Single
  Metri = IardeAMetri (CSng(txtIarde.Text))
  txtMetri.Text = CStr(Metri)
End Sub
```

Questa procedura di gestione degli eventi dice: "Quando l'utente fa clic su un pulsante di comando denominato cmdConversione, fai quanto segue".

1. **Crea una variabile denominata `Metri` e definiscila in modo che contenga solo dati di tipo `Single`.**
2. **Prendi il valore memorizzato nella proprietà `Text` della casella di testo denominata `txtIarde` dopo averlo convertito in un valore numerico utilizzato come argomento per la funzione `IardeAMetri`.**
3. **La funzione `IardeAMetri` utilizza la conversione in tipo `Double` del valore contenuto in `txtIarde.Text` come argomento, lo moltiplica per**

0.9 e utilizza il nuovo risultato come valore di ritorno della funzione IardeAMetri. Tale risultato viene poi assegnato alla variabile Metri.

4. **Il valore memorizzato nella variabile Metri viene convertito in una stringa e inserito nella proprietà Text della casella di testo denominata txtMetri.**

Notate le differenze esistenti tra una chiamata di procedura e una chiamata di funzione; quando si chiama una procedura è possibile utilizzare uno dei tre seguenti metodi.

- ✓ `NomeProcedura ElencoArgomenti`
- ✓ `NomeProcedura (ElencoArgomenti)`
- ✓ `Call NomeProcedura (ElencoArgomenti)`

È disponibile un unico sistema per chiamare una funzione:

```
Variabile = NomeFunzione(ElencoArgomenti)
```

Dato che il nome della funzione rappresenta un valore singolo, è possibile utilizzare il nome della funzione in qualsiasi espressione matematica, come:

```
Variabile = NomeFunzione(ElencoArgomenti) + Variabile
```

Quindi una procedura che chiama una funzione chiamate IardeAMetri potrebbe assomigliare a quanto segue:

```
Private Sub cmdStranaConversione_Click()
  Dim NuovoValore As Single, NuovoValore As Single
  NuovoValore = IardeAMetri (Metri) + 32) * 4
End Sub
```

Determinare il tipo di dati del valore restituito

Dato che una funzione restituisce un valore, potete specificare di quale tipo di dati quel valore deve essere.

Osservate per esempio il seguente codice:

```
Public Function IardeAMetri (Iarde) As Single
Const Conversione = 0.9
  IardeAMetri = Iarde * Conversione
End Function
```

Questo codice definisce il valore di `IardeAMetri` come tipo di dati `Single`. Potete definire una funzione in modo che il valore restituito sia di uno dei seguenti tipi di dati.

- ✓ Integer.
- ✓ Long.
- ✓ Single.
- ✓ Double.
- ✓ Currency.
- ✓ String.

A prescindere dal tipo di dati associato al valore restituito, la variabile alla quale si vuole assegnare questo valore dovrà contenere lo stesso tipo di dati. Per esempio:

```
Public Function IardeAMetri (Iarde) As Single
Const Conversione = 0.9
  IardeAMetri = Iarde * Conversione
End Function

Private Sub cmdConversione_Click()
Dim Metri As Single
  Metri = IardeAMetri (CSng(txtIarde.Text))
  txtMetri.Text = CStr(Metri)
End Sub
```

In questo esempio, la variabile `Metri` viene definita come tipo di dati `Single` e la funzione `IardeAMetri` utilizza lo stesso tipo di dati.

Se la variabile `Metri` fosse stata definita in questo modo:

```
Dim Metri As String
```

la riga

```
Metri = IardeAMetri (CSng(txtIarde.Text))
```

non avrebbe funzionato perché `Metri` è un tipo di dati String, mentre `IardeAMetri` restituisce dati di tipo `Single`. Dato che, `Metri` attende una stringa, mentre `IardeAMetri` le fornisce un numero, il programma non funziona.

Definire il tipo di dati degli argomenti

Gli argomenti sono dati (numeri, stringhe o variabili che rappresentano un numero o una stringa) di cui la funzione ha bisogno per lavorare.

Oltre a definire il numero degli argomenti in un elenco, si può specificare il tipo di dati che ciascun argomento deve rappresentare.

Per esempio, è possibile definire un argomento in modo che rappresenti uno dei seguenti tipi.

- ✓ Integer.
- ✓ Long.
- ✓ Single.
- ✓ Double.
- ✓ Currency.
- ✓ String.

Per definire il tipo di un argomento, dovete dichiararlo nell'elenco degli argomenti:

```
Public Function Conversione(Fahrenheit As Integer, Celsius As
Integer)
```

Questo esempio definisce due argomenti entrambi di tipo di dati Integer. Quanto segue mostra l'unica chiamata di procedura che può funzionare:

```
Private Sub cmdTest_Click()
Dim X, Y, Z As Integer
Dim A, B, C As String
Dim L, M, N As Single
  Z = Conversione(X, Y)    'Questa funziona
  C = Conversione(A, B)    'A e B non sono Integer
  L = Conversione(M, N)    'M e N non sono Integer
  Z = Conversione("Ciao", X)    '"Ciao" non è un Integer
End Sub
```

La ragione principale per specificare il tipo di argomento è impedire che una funzione utilizzi il tipo di dati sbagliato. Per esempio, se la funzione si attendesse un intero, ma ottenesse una stringa, il programma potrebbe non funzionare. Peggio ancora, potrebbe creare un errore logico. Sebbene non sia obbligatorio dichiarare il tipo di argomento, sarebbe meglio farlo sempre per evitare errori.

Problemi nel passare degli argomenti

Possono verificarsi due tipi di problemi quando si chiamano le funzioni; il primo si ha quando il numero degli argomenti non corrisponde al numero degli argomenti specificati dalla funzione.

Un altro problema si verifica quando i tipi di argomenti non corrispondono ai tipi di argomenti specificati dalla funzione.

Mettete alla prova le vostre nuove conoscenze

1. **Quando utilizzereste una funzione e quando una procedura?**

   a. Nel Capitolo 27 abbiamo imparato a conoscere le procedure, quindi è lì che dobbiamo utilizzarle. In questo capitolo si utilizzano solo funzioni perché questo è l'argomento della lezione.

   b. Le funzioni servono per calcolare un unico valore, le procedure per modificare gli argomenti.

   c. Le funzioni e le procedure sono identiche, tranne per il fatto che utilizzano nomi diversi, hanno scopi differenti e non hanno lo stesso aspetto.

   d. Si utilizza una funzione soltanto se non si ha il coraggio di utilizzare una procedura, come farebbero i veri programmatori.

2. **Quale riga è una chiamata di funzione e quale una chiamata di procedura?**

```
Private Sub cmdVisualizza()

Dim Nino, Pietro, Giorgio As
Double

Pietro = 3

Giorgio = 0

Nino = AranciaMeccanica(Pietro,
Giorgio) ' Riga 5

ComportamentoCondizionato(Nino)
' Riga 6

End Sub
```

   a. La quinta riga è una chiamata di funzione, dato che il nome della funzione viene assegnato ad una variabile, mentre la sesta riga è una chiamata di procedura, dato che non vi sono assegnamenti.

   b. Sia la quinta che la sesta riga sono chiamate di procedure perché sono convinto che questa sia una domanda trabocchetto; inoltre il fatto di avere sbagliato il 99% delle risposte dall'inizio del libro non è ancora stato sufficiente per convincermi che la risposta corretta è sempre la più ovvia.

   c. La sesta riga è una chiamata di funzione perché è diversa dalla quinta riga, un'altra chiamata di funzione.

   d. Non so la risposta, ma questa domanda sembra strana quanto un'arancia meccanica. Avete mai letto il libro di Anthony Burgess con il finale originale?

Fornire il numero sbagliato di argomenti

Quando definite una funzione con un elenco, in quest'ultimo viene specificato il numero di argomenti che la funzione deve ricevere. Se chiamate questa funzione e non le passate il numero corretto, la funzione non potrà essere operativa. Per esempio:

```
Public Function Fiamme(Mail)
End Function
```

Questa funzione si aspetta un argomento che può essere un numero oppure una stringa. Nulla di quanto segue consente alla funzione di essere operativa, perché il numero degli argomenti è diverso da uno.

- ✓ `X = Fiamme`
- ✓ `X = Fiamme(9, "Taci!")`
- ✓ `X = Fiamme("Perché", "Vai via!", 4500, "OK")`

Fornire il tipo errato di argomenti

In modo analogo, quando chiamate una funzione, controllate sempre che gli argomenti possiedano lo stesso tipo di dati definito nell'elenco. Per esempio:

```
Public Function Fiamme(Mail As String)
End Function
```

Questa funzione si aspetta un argomento che deve essere di tipo String. Niente di quanto segue consente alla funzione di essere operativa, dato che gli argomenti non sono di tipo String.

- ✓ `X = Fiamme(78.909)`
- ✓ `X = Fiamme(9)`
- ✓ `X = Fiamme(34)`

Uscire anticipatamente da una funzione

Di solito una funzione viene eseguita fino al completamento di tutte le istruzioni. Tuttavia potreste voler uscire prima che sia terminata.

Per uscire in anticipo da una funzione, utilizzate il seguente codice:

```
Exit Function
```

Prima di uscire da una funzione, verificate di avere assegnato alla variabile corretta il valore che la funzione deve restituire; in caso contrario il programma potrebbe non funzionare nel modo corretto.

Ora provate voi a utilizzare le funzioni

Il seguente programma di esempio consente di esaminare il funzionamento di una funzione. In questo caso, la funzione converte le iarde in metri.

Se non avete voglia di digitare tutto, troverete il programma memorizzato sul CD-ROM sotto il nome FUNCTION.VBP.

Oggetto	*Proprietà*	*Impostazione*
Form	Caption	Converte le iarde in metri
Etichetta1	Caption	iarde
	Heigh	495
	Left	360
	Top	360
	Width	1215
CasellaTesto1	Height	495
	Left	1680
	Name	txtIarde
	Text	(vuoto)
	Width	1215
Etichetta2	Caption	metri
	Height	495
	Left	360
	Top	1320
	Width	1215
CasellaTesto2	Height	495
	Left	1680
	Name	txtMetri
	Text	(vuoto)
	Width	1215
Comando1	Caption	&Converti
	Height	495
	Left	840
	Name	cmdConverti
	Top	2400

Oggetto	*Proprietà*	*Impostazione*
	Width	1215
Comando2	Caption	Exit
	Height	495
	Left	2520
	Name	cmdEsci
	Top	2400
	Width	1215

Digitare quanto segue nella finestra Codice:

```
Public Function IardeAMetri(Iarde) As Single
Const Conversione = 0.9
  IardeAMetri = Iarde * Conversione
End Function

Private Sub cmdEsci_Click()
  Unload Me
End Sub

Private Sub cmdConverti_Click()
Dim Metri As Single
  Metri = IardeAMetri(CSng(txtIarde.Text)
  txtMetri.Text = CStr(Metri)
End Sub
```

Capitolo 30

La programmazione orientata agli oggetti

In questo capitolo

- Definizione di programmazione orientata agli oggetti
- Creare gli oggetti in Visual Basic
- Utilizzare gli oggetti di Visual Basic

Visual Basic, sempre al passo con le ultimissime mode nel campo della programmazione, vi offre la possibilità di utilizzare la programmazione orientata agli oggetti per sviluppare le vostre eccezionali applicazioni. Ovviamente, a meno che non sappiate che cos'è la programmazione orientata agli oggetti e che cosa questo tipo di programmazione può fare per voi, questa caratteristica recente è utile quanto dare una sega a una scimmia e aspettarsi che costruisca una capanna tra i rami di un albero.

Quindi, prima di entusiasmarvi per la programmazione orientata agli oggetti con Visual Basic, fareste meglio innanzitutto a scoprire che cosa dovrebbe fare questo tipo di programmazione per voi.

Che cos'è la programmazione orientata agli oggetti?

La programmazione è un'arte più che una scienza, il che significa che il fatto di avere una laurea in informatica non garantisce che si sia in grado di scrivere programmi migliori rispetto a quelli che potrebbe scrivere chiunque non abbia neanche concluso la scuola superiore.

Per contribuire alla trasformazione della programmazione in una scienza, il settore informatico sta cercando di sviluppare alcune linee guida in grado di aiutare a scrivere programmi il più velocemente e facilmente possibile. Il primo tentativo di

scoprire questo "Sacro Graal" della programmazione fu qualcosa definito *programmazione strutturata*, che incoraggiava a suddividere un programma in sottoprogrammi, dove ogni sottoprogramma eseguiva esattamente una funzione.

Sebbene la programmazione strutturata fosse in grado di aiutare le persone a creare programmi che erano più semplici da scrivere e da modificare, rimaneva tuttavia un problema. Spesso diverse parti di un programma avevano accesso agli stessi dati, per esempio un file memorizzato su disco. Questo significava che se si voleva cambiare il modo in cui un programma aveva accesso a dati specifici, era necessario eseguire una ricerca esaustiva in tutto il programma e trovare tutti i comandi che avevano accesso a quei dati. Bastava non individuare un comando per introdurre un errore nel programma. Nella Figura 30.1 viene illustrata la differenza tra il funzionamento di un programma orientato agli oggetti e un programma non orientato agli oggetti; esaminate quella figura se pensate che un'immagine valga più di mille parole.

Ora, l'ultima novità della scienza informatica riguarda la *programmazione orientata agli oggetti* (object-oriented programming), spesso abbreviata in OOPs, come "Oops, hai sprecato quattro anni di università studiando la programmazione strutturata".

Il concetto fondamentale della programmazione orientata agli oggetti è la suddivisione del programma in parti isolate denominate *oggetti* (che altro?). Ogni oggetto contiene due elementi.

- ✓ Dati (spesso denominati proprietà).
- ✓ Comandi per gestire quei dati (spesso denominati *metodi*).

Quando un programma vuole accedere ai dati, deve semplicemente dare un comando all'oggetto che li contiene. Il programma principale non accede mai direttamente ai dati ma dà i comandi all'oggetto per accedere a essi.

Programma non orientato agli oggetti
Comandi per aprire un file
Comandi per visualizzare una finestra
Comandi per ridimensionare una finestra
Comandi per rinominare un file
Comandi per chiudere una finestra
Comandi per salvare un file

I comandi BASIC possono essere memorizzati in qualunque punto del programma, e ciò rende difficile all'inizio sapere dove cercare.

Programma orientato agli oggetti
(Oggetto File) Comandi per aprire un file Comandi per rinominare un file Comandi per salvare un file
(Oggetto Finestra) Comandi per visualizzare una finestra Comandi per ridimensionare una finestra Comandi per chiudere una finestra

I comandi BASIC sono isolati negli oggetti, che sono facili da individuare e poi modificare.

Figura 30.1 Come la programmazione orientata agli oggetti può rendere il programma più facile da modificare

Con questa tecnica, se dovete modificare il modo in cui il programma accede a dati specifici, non dovete far altro che cambiare i comandi all'interno di quell'oggetto invece di andare a toccare i comandi in tutto il programma. In questo modo, la programmazione orientata agli oggetti permette di isolare i comandi che influiscono su dati specifici, riducendo di conseguenza la possibilità di introdurre nuovi errori nel programma quando è necessario modificarlo.

Si dice che la programmazione orientata agli oggetti renda più semplice la stesura, la modifica e l'utilizzo dei programmi. In realtà, la programmazione orientata agli oggetti da sola non è in grado di rendere i vostri programmi utili o privi di errori. Un programmatore incapace che utilizza la programmazione a oggetti continuerà a essere meno efficiente di un programmatore eccellente che non si avvale della programmazione orientata agli oggetti. Non lasciatevi indurre a credere che i vostri programmi saranno più veloci, funzioneranno meglio o venderanno più copie solo perché avete utilizzato la programmazione orientata agli oggetti.

Concetti teorici sui moduli di classe

Ora che ci si è fatta un'idea generale su come funzionano gli oggetti, si potrebbe avere voglia di sapere come gli oggetti funzionano in Visual Basic. Per creare un oggetto in Visual Basic, dovete creare qualcosa definito *modulo di classe* (class module), che ha l'estensione di file .CLS, come NOCLASS.CLS.

Il modulo di classe definisce i seguenti dati e comandi.

- ✓ Il tipo di dati che un oggetto può contenere (ma non i dati veri e propri).
- ✓ I comandi BASIC (i metodi) per la gestione dei dati in quel determinato oggetto.

I dati definiti in un oggetto vengono denominati *proprietà*, mentre i comandi BASIC che gestiscono i dati in un oggetto vengono definiti *metodi*. Se i termini proprietà e metodi non vi chiariscono le idee non preoccupatevi, non siete i soli ad avere questa sensazione.

Un modulo di classe agisce come uno stampino per biscotti: dato che questo tipo di modulo definisce l'oggetto ma non contiene dati, proprio come uno stampino dà la forma a un biscotto.

Un tipico modulo di classe consiste di tre parti.

- ✓ Le dichiarazioni delle variabili.
- ✓ Le dichiarazioni delle proprietà.
- ✓ I metodi, ossia i comandi BASIC, memorizzati nelle procedure che gestiscono le variabili e le proprietà.

Nel codice del modulo di classe d'esempio riportato di seguito, viene dichiarata una sola variabile utilizzando una linea di codice di questo genere: Private mvar-Messaggio As String.

La dichiarazione della prima proprietà comincia con la riga "Public Property Let Messaggio (ByVal vDati As String)" e la dichiarazione della seconda proprietà comincia con la riga "Public Property Get Messaggio() As String".

Il metodo comincia con la riga "Public Sub Correggi()".

```
Private mvarMessaggio As String 'copia locale

Public Property Let Messaggio(ByVal vDati As String)
  mvarMessaggio = vDati
End Property

Public Property Get Messaggio() As String
  Messaggio = mvarMessaggio
End Property

Public Sub Correggi()
  Dim NuovaStringa As String, StringaIniziale As String
    StringaFinale As String
  Dim Posizione As Integer, SceglineUno As Integer
  NuovaStringa = ""
  StringaFinale = ""
  SceglineUno = CInt((4 * Rnd) + 1)
  Select case SceglineUno
    Case 1
      NuovaStringa = "fabbricare con standard
        di qualità elevati"
    Case 2
      NuovaStringa = "consegnare con rapidità
        in tutto il mondo"
    Case 3
      NuovaStringa = "salvaguardare la qualità del prodotto"
    Case Else
      NuovaStringa = "personalizzare la funzionalità"
End Select

Posizione = InStr(mvarMessaggio, "fare")
If Location = 0 Then
  mvarMessaggio = "Dovete elaborare un discorso che contenga
        la parola 'fare'."
  Else
    StringaIniziale = Left(mvarMessaggio, Posizione -1)
    StringaFinale = Right(mvarMessaggio, Len(mvarMessaggio) -
(Posizione + 4)
    mvarMessaggio = StringaIniziale & NuovaStringa &
```

```
StringaFinale
  End If
End

Sub
```

Dichiarare le variabili

Sarebbe meglio dichiarare tutte le variabili utilizzate dalla vostra classe all'inizio di questa, in modo da sapere quale tipo di informazioni si stanno utilizzando. Se si vuole dichiarare una variabile che qualunque parte del programma può utilizzare per memorizzare o recuperare le informazioni, è possibile utilizzare una variabile pubblica nel seguente modo:

```
Public SoldiRubati As Currency
```

Se dichiarate una variabile pubblica, qualunque parte del programma (compresi altri oggetti) può inserire i dati nella variabile, il che significa che il debugging può essere estremamente difficile. In linea di principio, comunque, è preferibile non utilizzare variabili pubbliche a meno che non si abbia una valida ragione per farlo.

Se volete dichiarare una variabile che soltanto la vostra classe può utilizzare, potete dichiarare una variabile privata nel seguente modo:

```
Private Contatore As Integer
```

Definire le proprietà di un oggetto

Per quanto riguarda la parte restante del vostro programma Visual Basic, esistono due tipi di proprietà.

- ✓ Le proprietà alle quali il programma può attribuire un valore.
- ✓ Le proprietà dalle quali il programma può recuperare un valore.

Se avete, per esempio, una proprietà denominata `Direzione` alla quale volete attribuire un valore, dovete dichiarare la proprietà `Direzione` nella relativa classe, come segue:

```
Private mStrada As Integer
Property Let Direzione(ByVal QualeStrada As Integer)
  mStrada = QualeStrada
End Property
```

Ecco come Visual Basic interpreta questo codice.

1. La prima riga definisce una variabile privata, denominata `mStrada`, che può contenere un valore di tipo Integer.
2. La seconda riga dice a Visual Basic: "Lascia che sia un'altra parte del programma ad attribuire un valore di tipo Integer alla variabile `mStrada`". Per attribuire un valore alla proprietà di un oggetto, utilizzate un codice simile al seguente:

   ```
   Set m_Object = New cObject
   m_Object.Direzione = 5
   ```

   Queste due righe di codice dicono a Visual Basic di creare un oggetto, utilizzando le parole chiave `Set` e `New` (queste parole chiave vengono illustrate dettagliatamente più avanti in questo capitolo), e di attribuire un valore alle proprietà dell'oggetto, proprio come si assegna una proprietà a una casella di testo o a un'etichetta su una form.
3. La terza riga dice a Visual Basic: "Qualunque valore venga attribuito alla proprietà `Direzione`, devi memorizzarlo nella variabile privata `mStrada`".
4. La quarta riga dice a Visual Basic: "Questa è la fine della definizione della proprietà che permette a un'altra parte del programma di attribuire un valore a una proprietà".

Per recuperare le informazioni intrappolate nella proprietà di un oggetto, dovete dichiarare la proprietà `Direzione` nella relativa classe, come segue:

```
Property Get Direzione() As Integer
  Direzione = mStrada
End Property
```

Ecco come Visual Basic interpreta questo codice.

1. La prima riga dice a Visual Basic: "La proprietà `Direzione` di questo oggetto può contenere soltanto valori di tipo Integer".
2. La seconda riga dice a Visual Basic: "Attribuisci il valore memorizzato nella variabile privata `mStrada` alla proprietà `Direzione`".
3. La terza riga dice a Visual Basic: "Questa è la fine della definizione della proprietà che permette a un'altra parte del programma di ottenere un valore dalla proprietà".

Per recuperare effettivamente le informazioni memorizzate nella proprietà di un oggetto, potete utilizzare il seguente codice BASIC:

```
CodiceStrada = mObject.Direzione
```

Notate la sottile differenza tra le due definizioni di proprietà. Per permettere a un'altra parte del programma di definire un valore per una proprietà dell'oggetto, si devono eseguire i passi elencati.

- ✓ Utilizzare la magica parola chiave `Let`.
- ✓ Definire una variabile e il suo tipo di dati racchiudendoli tra parentesi, come `(QualeStrada As Integer)`.
- ✓ Attribuire una variabile privata (dichiarata in precedenza nel modulo di classe) che sia uguale al valore della variabile identificata tra parentesi, come `mStrada = QualeStrada`.

Invece per far sì che un'altra parte del programma recuperi (`Get`) un valore da una proprietà dell'oggetto, si devono eseguire i passi elencati.

- ✓ Utilizzare la magica parola chiave `Get` per attribuire un tipo di dati alla proprietà.
- ✓ Lasciare vuote le parentesi, come `()`.
- ✓ Assegnare il valore di una variabile privata al nome della proprietà (come `Direzione`), per esempio `Direzione = mStrada`.

Le istruzioni `Property Let` e `Property Get` devono essere utilizzate per ogni proprietà dell'oggetto.

Scrivere i metodi di un oggetto

Dopo che si saranno dichiarate le variabili di cui un oggetto potrebbe aver bisogno e si saranno definite le proprietà dell'oggetto, dovranno essere scritti i metodi (procedure o funzioni) che svolgono qualche operazione con i dati memorizzati all'interno dell'oggetto.

Scrivere una procedura o una funzione è abbastanza semplice (si vedano i Capitoli 27, 28 e 29); la differenza principale è che invece di utilizzare la parola chiave `Private` davanti alle dichiarazioni di procedure o di funzioni, dovete utilizzare la parola chiave `Public` nel seguente modo:

```
Public Sub Sposta ()
  'Qui viene inserito il codice BASIC
End Sub
```

oppure

```
Public Function XPosizione () As Integer
  'Qui viene inserito il codice BASIC
End Function
```

Quando volete chiamare i metodi di un oggetto, utilizzate il nome dell'oggetto insieme al metodo, come segue:

```
m_Object.Sposta
```

Il codice precedente dice a Visual Basic: "Trova un oggetto denominato `m_Object` e chiama la procedura `Sposta`". Il vostro programma principale non avrà la minima idea di come funzioni la procedura `Sposta` fino a quando non avrete immesso il relativo codice.

Progettare una classe sulla carta

Anche se potreste lanciarvi a capofitto nella scrittura del codice BASIC che implementa un modulo di classe, sarebbe meglio, prima, prendersi un po' di tempo per progettarlo. Qual è il metodo migliore da seguire? Nessuno (non siete contenti di avere acquistato un libro che vi ha detto questo?).

In realtà, il design ottimale per un modulo di classe dipende da come prevedete di utilizzarlo. Prendete in considerazione i suggerimenti di seguito riportati per avere alcune indicazioni nella progettazione delle classi.

- Per determinare le proprietà di un modulo di classe, decidete il blocco di dati fondamentale che il programma deve gestire. Se si sta scrivendo un programma per memorizzare le informazioni sui dipendenti, il modulo di classe deve comprendere le proprietà che contengono nomi, indirizzi, numeri telefonici, QI o precedenti penali. Se si sta scrivendo un videogame in cui gli alieni balzano fuori sullo schermo in modo che si possa colpirli, il modulo di classe deve contenere le coordinate X e Y della posizione dell'alieno sullo schermo.
- Per determinare i metodi del modulo di classe, bisogna prima di tutto stabilire che cosa il programma principale Visual Basic deve fare con le informazioni memorizzate. Per esempio, un modulo di classe che contiene i nomi e gli indirizzi degli impiegati potrebbe aver bisogno di metodi che permettono al programma principale di ricercare, ordinare e stampare i dati degli impiegati. In modo analogo, un modulo di classe che contiene le coordinate X e Y dell'alieno potrebbe aver bisogno di metodi che permettono al programma principale di spostare, visualizzare e far andare in pezzi l'alieno.
- Dopo avere abbozzato il tipo di proprietà e di metodi che il vostro modulo di classe deve comprendere, siete pronti a crearne uno effettivo.

Creare un modulo di classe con il programma Creazione guidata classi VB

Dato che la creazione delle classi può essere difficile se non si sa quello che si sta facendo, Visual Basic offre un utile programma denominato Creazione guidata classi VB, che permette di definire i metodi e le proprietà della classe e poi si occupa di scrivere il codice BASIC al posto vostro.

Ovviamente, se si preferisce la strada più difficile, è possibile creare i moduli di classe partendo da zero, ma perlomeno grazie al programma Creazione guidata classi VB non si è costretti a farlo se non si vuole perdere troppo tempo.

Invece di creare un modulo di classe in fretta e furia, sarebbe meglio progettare innanzitutto proprietà e metodi dell'oggetto sulla carta. Solo dopo che avrete raggiunto gli obiettivi desiderati, potrete buttarvi a capofitto nella creazione del modulo di classe.

Creare una nuova classe

È necessario creare un modulo di classe separato per ogni oggetto che sarà utilizzato nel programma. Per la creazione (o per modificare un modulo già esistente), utilizzate la procedura di seguito riportata.

1. **Selezionate Progetto-Inserisci modulo di classe.**

   Appare una finestra di dialogo Inserisci modulo di classe, come illustrato nella Figura 30.2.

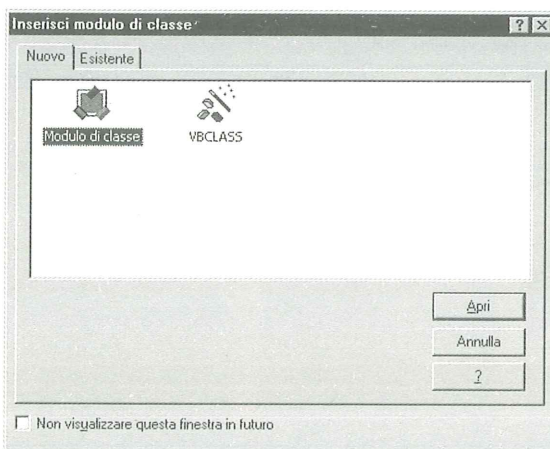



Figura 30.2 La finestra di dialogo Inserisci modulo di classe per la creazione di una classe

2. **Fate clic sull'icona Creazione guidata classi VB e poi fate clic su Apri.**

   Appare la finestra Creazione guidata classi, come illustrato nella Figura 30.3.

3. **Selezionate File-Nuovo-Classe oppure fate clic sull'icona Aggiungere una nuova classe nella barra degli strumenti.**

   Appare la finestra Creazione modulo di classe, illustrata nella Figura 30.4. Se volete soltanto modificare un modulo di classe esistente, saltate le fasi 3 e 4 e fate semplicemente clic sul nome di classe visualizzato nella finestra.

4. **Digitate un nome per la classe nella casella di testo Nome e poi fate clic su OK.**

   Se non scegliete un nome, Visual Basic attribuirà alla vostra classe un nome tanto generico e insulso come Class1.

La procedura appena descritta permette soltanto di creare e di attribuire un nome ad un modulo di classe, ma vanno ancora definiti proprietà e metodi all'interno di essa.

Creare le proprietà per le vostre classi

Per definire le proprietà di un modulo di classe, utilizzate la procedura di seguito riportata (partendo dal presupposto che la finestra Creazione guidata classi sia già visualizzata).

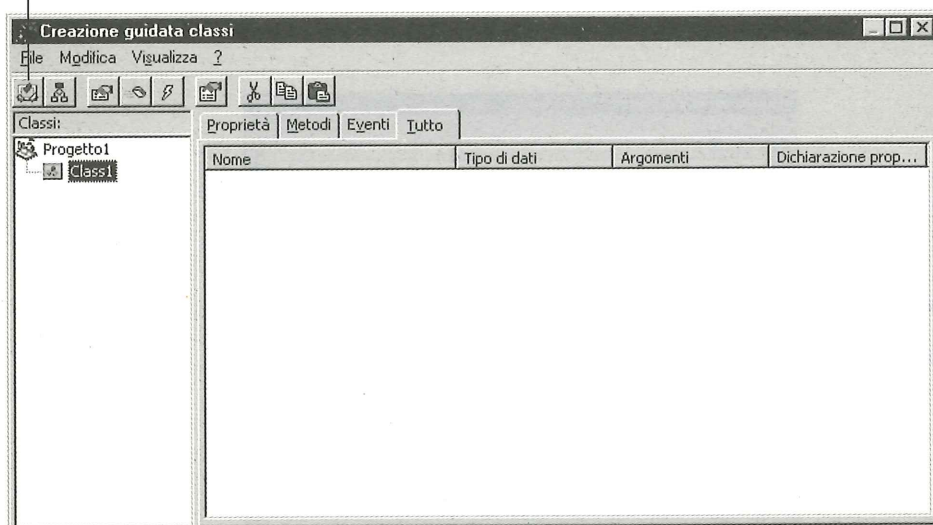



Figura 30.3 La finestra Creazione guidata classi VB per la creazione delle classi

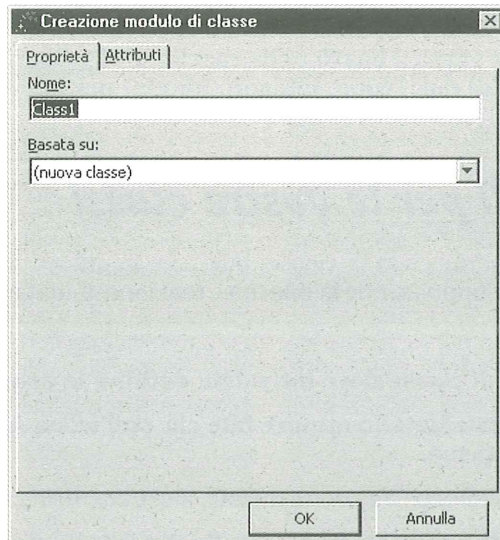



Figura 30.4 La finestra di dialogo Creazione modulo di classe per attribuire un nome alla classe

1. **Fate clic sul modulo di classe dove intendete definire le proprietà.**
2. **Selezionate File-Nuovo-Proprietà oppure fate clic sull'icona Aggiungi una nuova proprietà alla classe.**

   Appare la finestra di dialogo Creazione proprietà, illustrata nella Figura 30.5.

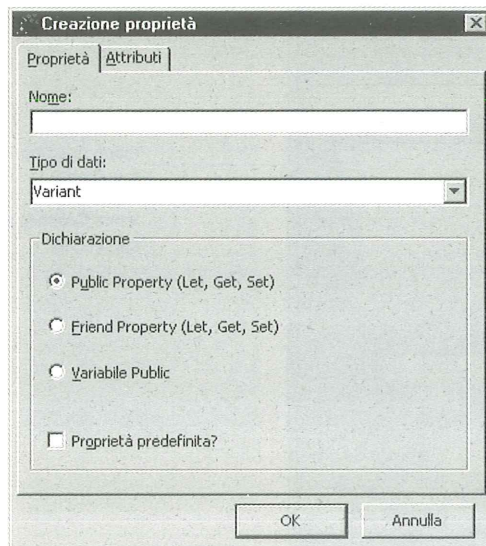



Figura 30.5 La finestra di dialogo Creazione proprietà per definire le proprietà di un oggetto

3. **Digitate il nome della proprietà nella casella di testo Nome.**
4. **Fate clic sulla freccia verso il basso nella casella di riepilogo Tipo di dati e selezionate il tipo di dati (come Integero Single) per la proprietà.**
5. **Fate clic su OK.**

Creare i metodi per le vostre classi

Per definire i metodi per un modulo di classe, utilizzate la procedura di seguito riportata (partendo dal presupposto che la finestra Creazione guidata classi sia già visualizzata).

1. **Fate clic sul modulo di classe dove intendete definire le proprietà.**
2. **Selezionate File-Nuovo-Metodo oppure fate clic sull'icona Aggiunge un nuovo metodo alla classe.**

   Appare la finestra di dialogo Creazione metodi, illustrata nella Figura 30.6.
3. **Digitate il nome della proprietà nella casella di testo Nome.**
4. **Fate clic sull'icona Aggiunge un nuovo argomento (il segno più) se i metodi necessitano di argomenti.**

   Appare la finestra di dialogo Aggiungi argomento, illustrata nella Figura 30.7.
5. **Digitate un nome per l'argomento nella casella di testo Nome.**

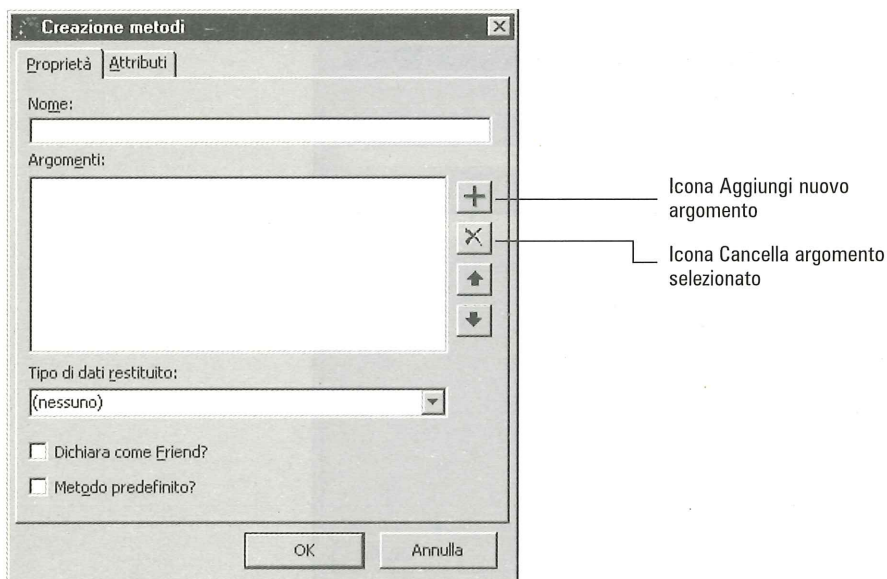



Figura 30.6 La finestra di dialogo Creazione metodi per definire i metodi di un oggetto

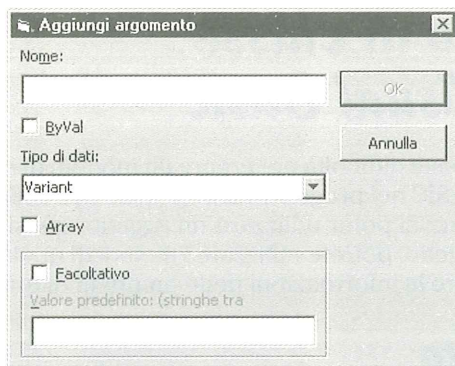



Figura 30.7 La finestra di dialogo Aggiungi argomento

6. **Fate clic sul tipo di dati (per esempio Integer o Currency) nella casella di riepilogo Tipo di dati.**
7. **Fate clic su OK.**
8. **Fate clic sulla freccia verso il basso nella casella di riepilogo Tipo di dati restituito e selezionate un tipo di dati.**

   Saltate questa fase se non volete che il metodo restituisca un valore.
9. **Fate clic su OK.**

Uscire dal programma Creazione guidata classi

Quando avete terminato di creare o modificare i moduli di classe, potete liberarvi della finestra Creazione guidata classi utilizzando uno di questi metodi.

- ✓ Fate clic sulla casella di chiusura nella finestra Creazione guidata classi.
- ✓ Selezionate File-Esci.

Se viene visualizzata una finestra di dialogo che vi chiede se volete aggiornare il progetto Visual Basic, fate clic su Sì. Dopo che la finestra Creazione guidata classi è scomparsa dallo schermo, si può fare clic sul modulo di classe appena creato e visualizzare il codice BASIC che Visual Basic ha gentilmente creato in modo automatico.

Dopo aver utilizzato il programma Creazione guidata classi per creare un modulo di classe, si deve ancora scrivere il codice all'interno del modulo di classe affinché questo possa fare qualcosa di utile. Dopodiché dovrete scrivere il codice nel programma principale per utilizzare il modulo di classe.

Utilizzare un modulo di classe in un programma Visual Basic

Dopo avere affrontato tutte queste difficoltà per creare un modulo di classe, dovete ancora scrivere il codice BASIC nel programma principale per utilizzare effettivamente la classe creata. Prima di poter utilizzare un oggetto, questo va creato. Dopo che avrete creato un oggetto, potrete utilizzare i metodi di quell'oggetto oppure memorizzare o recuperare le informazioni nelle proprietà di quell'oggetto.

Creare un oggetto

Nonostante abbiate già affrontato tutte le difficoltà legate alla creazione e alla definizione del modulo di classe, dovete ancora creare un oggetto basato sul vostro modulo di classe. Nel mondo della programmazione orientata agli oggetti, la creazione di un oggetto viene detta "creazione di un'*istanza*" (instance). Anche in questo caso, potete notare che gli scienziati dei computer non sono più bravi nel scegliere termini autoesplicativi di altri esperti dei rispettivi campi.

Per creare l'istanza di un oggetto, bisogna creare un oggetto che rappresenti il modulo di classe utilizzando le parole chiave Set e New, come segue:

```
Set NomeOggetto = New NomeClasse
```

Ecco come Visual Basic interpreta questa singola riga di codice.

1. La parola chiave `Set` dice a Visual Basic: "Preparati a creare un oggetto".
2. La variabile `NomeOggetto` è il nome dell'oggetto.
3. La parola chiave `New` dice a Visual Basic: "Crea un nuovo oggetto in base al modulo di classe definito da `NomeClasse`".

Utilizzare un oggetto

Dopo avere creato un oggetto, la fase finale consiste nell'utilizzare quell'oggetto per i seguenti scopi.

- ✓ Inserire un valore nella proprietà di un oggetto.
- ✓ Recuperare un valore dalla proprietà di un oggetto.
- ✓ Utilizzare i metodi dell'oggetto per fare qualcosa con i dati in esso contenuti.

Per inserire un valore nella proprietà dell'oggetto, dovete utilizzare il seguente codice:

```
NomeOggetto.Proprietà = Valore
```

Per ottenere il valore memorizzato nella proprietà dell'oggetto, dovete utilizzare il seguente codice:

```
Variabile = NomeOggetto.Property
```

Per utilizzare un metodo dell'oggetto, dovete utilizzare il seguente codice:

```
NomeOggetto.Metodo
```

Provate anche voi

Naturalmente il metodo migliore per comprendere qualsiasi cosa è fare ciò che non si capisce da soli; quindi nel caso questo capitolo non vi sia sembrato chiaro, provate a creare il programma di seguito riportato, che dimostra come un programma principale possa creare un oggetto da un modulo di classe, chiamare un metodo dell'oggetto per gestire i dati e poi recuperare in seguito i dati.

In questo programma particolare, digitate il discorso che illustra lo scopo della vostra azienda nella casella di testo superiore, controllando che comprenda la parola "Fare". Sarà poi compito di questo programma sostituire la parola "fare" con un termine aziendale più accettabile.

Questo programma è memorizzato sul CD-ROM per cui non è necessario che lo digitiate tutto se non ne avete voglia. Basta che carichiate MISSION.VBP.

Se non riuscite a trarre alcun vantaggio da questo capitolo, ricordatevi semplicemente che la programmazione orientata agli oggetti dovrebbe aiutare a organizzare i programmi in modo che diminuiscano le possibilità di inserire errori nei programmi da scrivere o da modificare.

Oggetto	*Proprietà*	*Valore*
Form	Caption	Correttore di discorsi
	Height	3600
	Left	0
	Top	0
	Width	4800
Testo1	Height	615
	Left	480

(continua)

Oggetto	*Proprietà*	*Valore*
MultiLine	True	
	Name	txtInput
	TabIndex	0
	Text (vuoto)	
	Top	360
	Width	3850
Testo2	Height	615
	Left	480
	MultiLine	True
	Name	txtOutput
	Text	(vuoto)
	Top	1440
	Width	385
Comando1	Caption	&Crea
	Height	495
	Left	600
	Name	cmdCrea
	Top	2400
	Width	1575
Comando2	Caption	E&sci
	Height	495
	Left	2640
	Name	cmdEsci
	Top	2400
	Width	1575

Fate doppio clic su uno dei due pulsanti di comando sulla form e create le seguenti procedure di gestione degli eventi:

```
Private Sub cmdCrea_Click()
  Set DiscorsoAziendale = New CorrettoreDiscorsi
```

```
  DiscorsoAziendale.Messaggio = txtInput.Text
  DiscorsoAziendale.Correggi
  txtOutput.Text = DiscorsoAziendale.Messaggio
End sub

Private Sub cmdEsci_Click()
  Unload Me
End Sub
```

Create un modulo di classe separato, attribuire al modulo il nome CorrettoreDiscorsi, create una proprietò chiamata Messaggio e create un metodo chiamato Correggi. Dovrete digitare il codice nel metodo Correggi:

```
Private mvarMessaggio As String 'copia locale

Public Property Let Messaggio(ByVal vDati As String)
  mvarMessaggio = vDati
End Property

Public Property Get Messaggio() As String
  Messaggio = mvarMessaggio
End Property

Public Sub Correggi()
  Dim NuovaStringa As String, StringaIniziale As String
    StringaFinale As String
  Dim Posizione As Integer, SceglineUno As Integer
  NuovaStringa = ""
  StringaFinale = ""
  SceglineUno = CInt((4 * Rnd) + 1)
  Select case SceglineUno
    Case 1
      NuovaStringa = "fabbricare con standard
        di qualità elevati"
    Case 2
      NuovaStringa = "consegnare con rapidità in tutto
        il mondo"
    Case 3
      NuovaStringa = "salvaguardare la qualità del prodotto"
    Case Else
      NuovaStringa = "personalizzare la funzionalità"
  End Select

Posizione = InStr(mvarMessaggio, "fare")
If Location = 0 Then
  mvarMessaggio = "Dovete elaborare un discorso
        che contenga la parola 'fare'."
  Else
```

```
    StringaIniziale = Left(mvarMessaggio, Posizione -1)
    StringaFinale = Right(mvarMessaggio, Len(mvarMessaggio)
        - (Posizione + 4)
    mvarMessaggio = StringaIniziale & NuovaStringa &
StringaFinale
  End If
End Sub
```

Capitolo 31

Gestione dei file

In questo capitolo

- Determinare come Visual Basic memorizza un programma
- Giocare con la finestra Gestione progetti
- Aggiungere file form, modulo e classe
- Inserire i controlli ActiveX nel programma

Durante i bei vecchi tempi, un singolo programma era costituito da un unico file. Era sufficiente modificare quel file per modificare l'intero programma. Tuttavia, i programmi sono diventati più complicati e ora sono costituiti da due o più file. Questo capitolo spiega in che modo Visual Basic gestisce i file multipli in modo che voi possiate non perdere il filo.

Come Visual Basic memorizza un programma

Quando si utilizza un programma di elaborazione testi, è possibile digitare una frase o trecento pagine di testo e il programma è in grado di memorizzare tutto in un unico file. Visual Basic invece salva sempre (sì, sempre) il programma in due o più file separati. Nella Tabella 31.1 sono elencati i tipi più comuni di file di Visual Basic che si possono incontrare insieme alla relativa estensione di tre lettere.

Ogni programma Visual Basic contiene un file di progetto (.VBP), che contiene un elenco di tutti i file Form, file Module, controlli ActiveX e file Class) che, uniti, compongono un programma Visual Basic.

Un *file form* è costituito da una finestra che rappresenta l'interfaccia utente del vostro programma e dal codice BASIC che dice a ogni pulsante (casella di controllo e così via) presente sulla finestra come comportarsi. La maggior parte dei programmi Visual Basic sono costituiti da uno o più file Form (FRM).

Tabella 31.1 File tipici di Visual Basic e relative estensioni di file

Tipo di file	*Estensione di tre lettere*	*Nome file d'esempio*
File progetto	.VBP	VIRUS.VBP
File form	.FRM	MENUPRIN.FRM
File modulo	.BAS	LIBRARY.BAS
Controllo ActiveX	.OCX	RICHTX32.OCX
File classe	.CLS	OGGETTI.CLS

Un file modulo contiene il codice BASIC che esegue un qualche tipo di calcolo indipendente dall'interfaccia utente; i programmi Visual Basic possono contenere zero o più file modulo (.BAS).

Un controllo ActiveX è un programma in miniatura che potete collegare al vostro programma per aggiungere alcune caratteristiche. Alcuni dei più noti controlli ActiveX permettono di aggiungere velocemente funzioni dei programmi di elaborazione testi, di grafica o di fogli elettronici ai programmi Visual Basic. Potete scegliere se acquistare i controlli ActiveX o scriverli da soli. Un programma Visual Basic può comprendere zero o svariati controlli ActiveX (.OCX).

Un file classe contiene il codice BASIC che definisce le diverse classi utilizzate dal programma Visual Basic. Una classe non è altro che un metodo divertente per organizzare i dati utilizzati dal programma (per avere ulteriori informazioni sulle classi, consultate il Capitolo 30). Un programma Visual Basic può includere zero o numerosi file Class (.CLS).

È possibile memorizzare i file Project in una cartella e tutti gli altri file (Form, Module, Class e Active) in cartelle completamente diverse. In ogni caso, se non tutti i file di un programma vengono memorizzati nella stessa cartella, potrebbe risultare difficile apportare modifiche in un momento successivo qualora non doveste riuscire a trovare tutti i file che compongono il programma.

File di progetto VBP

I file di progetto VBP contengono un elenco di tutti i file FRM, BAS, CLS e OCX che compongono il programma. La finestra Gestione progetti visualizza tutti i file memorizzati in un file di progetto VBP, come illustrato nella Figura 31.1.

Figura 31.1 La finestra Gestione progetti visualizza un elenco dei file che compongono un unico programma Visual Basic

Per facilitare l'organizzazione dei file, la finestra Gestione progetti li visualizza come se fossero memorizzati all'interno di cartelle separate.

- ✓ Form.
- ✓ Moduli.
- ✓ Moduli di classe.

La finestra Progetto non elenca tutti i controlli ActiveX utilizzati dal vostro programma in Visual Basic. Per vedere un elenco dei controlli ActiveX utilizzati da un programma, premete Ctrl+T.

Se non si vogliono vedere tutti i file presenti all'interno di una cartella, bisogna fare doppio clic sulla cartella per far apparire un'icona del segno più sulla sinistra. Poi, quando si vuole visualizzare un elenco di tutti i file di una particolare cartella, si deve fare doppio clic su quella cartella per visualizzare nuovamente i file.

Per creare un nuovo file di progetto VBP, si utilizzi la procedura di seguito riportata.

1. **Selezionate File-Nuovo progetto oppure premere CTRL+N.**

   Visual Basic visualizza la finestra di dialogo Nuovo progetto.

2. **Fate clic sul tipo di programma che volete creare (per esempio Standard EXE oppure ActiveX DLL) e poi fate clic su OK.**

Per caricare un file di progetto esistente, utilizzate la seguente procedura.

1. **Selezionate File-Apri progetto oppure premete CTRL+O oppure fate clic sull'icona Apri progetto.**

   Visual Basic visualizza la finestra di dialogo Apri progetto.

2. **Digitate il nome del progetto oppure fate clic sul file VBP che volete caricare e poi fate clic su Apri.**

Ogni volta che caricate un file di progetto VBP, Visual Basic carica automaticamente tutti i file elencati nel file VBP.

Per salvare il progetto Visual Basic per sempre (o almeno finché si cancella dal disco fisso), si utilizzi la procedura di seguito riportata.

1. **Selezionate File-Salva progetto oppure fate clic sull'icona Salva progetto nella barra degli strumenti.**

   Se non avete ancora salvato i file (quali i file form FRM o i file modulo BAS), viene visualizzata una finestra di dialogo Salva con nome.

2. **Digitate il nome dei file form FRM e fate clic su Salva.**

   Dopo che avrete salvato tutti i file form, Visual Basic visualizza la finestra di dialogo Salva progetto come.

3. **Digitate il nome del vostro file di progetto VBP e poi fate clic su Salva.**

Prima di provare a eseguire il programma, è bene accertarsi di avere salvato tutti i file. Spesso un programma contiene errori che provocano un crash del computer, il che significa che se non si salva il programma prima di eseguirlo, c'è il rischio di perdere tutte le modifiche appropriate (sì, tutte).

Se volete che Visual Basic salvi automaticamente le modifiche prima che eseguiate il programma, utilizzate la procedura di seguito riportata.

1. **Selezionate Strumenti-Opzioni.**

   Appare la finestra di dialogo Opzioni.

2. **Fate clic sulla scheda Ambiente.**

3. **Fate clic sul pulsante di opzione Salva senza conferma o su quello Salva con conferma e poi fate clic su OK.**

Aggiungere file a un file di progetto

Più programmi Visual Basic scriverete, più determinate parti di questi programmi diventeranno utili per i programmi che potreste scrivere in futuro. Fortunatamente, Visual Basic permette di prelevare con facilità file FRM, BAS e CLS creati per un altro programma e di aggiungerli al nuovo programma.

Per aggiungere questi tipi di file a un file di progetto VBP, selezionate Progetto e poi selezionate il tipo di file che volete aggiungere come Aggiungi Form o Aggiungi modulo.

Se aggiungete un file Form, Module o Class tratto da un altro programma Visual Basic, assicuratevi di salvare il file sotto un altro nome. Se due o più programmi Visual Basic condividono lo stesso file Form, Module o Class, qualsiasi modifica apportata al file influenzerà anche tutti i programmi Visual Basic che lo utilizzano.

Eliminare file da un file di progetto

Talvolta potreste voler eliminare definitivamente un file da un file di progetto. Per eliminare un file FRM o BAS da un file di progetto VBP, utilizzate la procedura di seguito riportata.

1. **Premete CTRL+R, selezionate Visualizza-Gestione progetti oppure fate clic sull'icona Gestione progetti nella barra degli strumenti.**
2. **Fate clic sul file (FRM, BAS o CLS) che volete eliminare.**
3. **Selezionate Progetto-Rimuovi.**

In alternativa, al punto 2 è possibile selezionare il file con il pulsante destro del mouse e quindi usare il pulsante sinistro per selezionare il comando Remove.

Quando eliminate un file da un file di progetto, continuerà ad esserci traccia di esso sul dischetto o sul disco fisso. L'operazione di eliminazione si limita a dire a Visual Basic: "Vedi quel file laggiù? Non voglio che faccia parte di questo particolare programma, quindi toglilo, ma conservalo su disco nel caso voglia inserirlo di nuovo". Per cancellare o eliminare un file da dischetto o da disco fisso, bisogna utilizzare il programma Gestione risorse e far scomparire definitivamente il file.

Aggiungere (o eliminare) i controlli ActiveX

Un controllo ActiveX è un programma in miniatura che si può inserire nei programmi Visual Basic per fornire loro ulteriori funzioni senza dover scrivere del codice BASIC. Dato che i controlli ActiveX sono utili per scrivere programmi rapidamente e facilmente, i programmi Visual Basic di molti non sono altro che una massa grumosa di controlli ActiveX collegati tra di loro da una minima parte di codice BASIC.

Sebbene Visual Basic sia dotato di diversi controlli ActiveX, è possibile acquistarne molti altri mediante ordini postali o Internet. Bisogna solo stare attenti a non fare eccessivo affidamento sui controlli ActiveX. Se una società ha scritto un controllo ActiveX che non funziona correttamente, allora neanche il programma Visual Basic funzionerà correttamente.

Ogni volta che si aggiunge un controllo personalizzato a un file di progetto VBP, l'icona di quel controllo personalizzato viene visualizzata nella casella degli strumenti di Visual Basic.

Per aggiungere un controllo ActiveX a un file di progetto VBP, utilizzate la procedura di seguito riportata.

1. **Selezionate Progetto-Componenti oppure premete CTRL+T.**

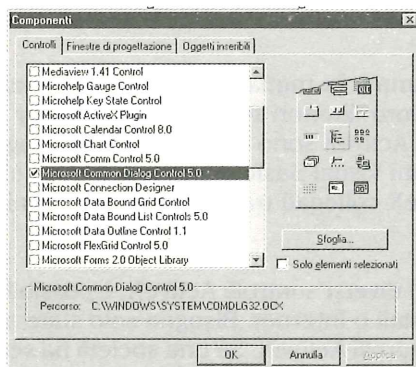
   Visual Basic visualizza la finestra di controllo Componenti, illustrata nella Figura 31.2.

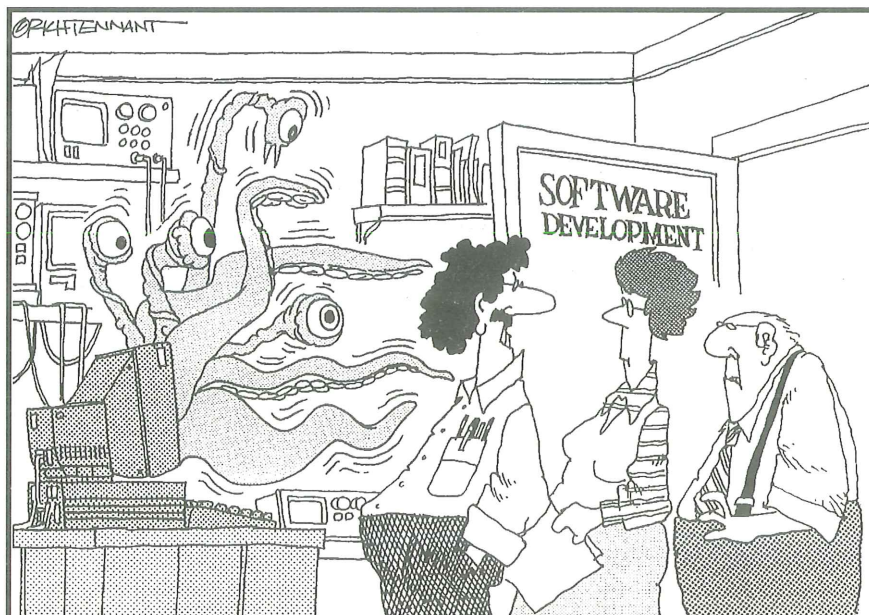
2. **Fate clic sulla casella di controllo del controllo ActiveX che volete utilizzare e poi fate clic su OK.**

   Visual Basic visualizza l'icona del controllo personalizzato nella casella degli strumenti di Visual Basic.

Dopo avere aggiunto un controllo ActiveX al programma, sarà necessario disegnarlo su una form e quindi modificarne le proprietà o scrivere il codice BASIC che permetta al controllo di svolgere qualche funzione.

Che ci crediate o meno, potete scrivere controlli ActiveX personalizzati utilizzando Visual Basic. (Spiego più dettagliatamente i controlli ActiveX nell'ultima edizione di MORE Visual Basic For Windows For Dummies, scritto dal vostro amato ed edito da IDG Books Worldwide, Inc). Mentre scrivete i controlli ActiveX tenete comunque presente che se create una funzione particolarmente utile nel vostro programma e ritenete che anche altri programmatori potrebbero trovare utile, potete salvare la funzione sotto forma di controllo ActiveX e distribuire o vendere le copie ad altri.

Figura 31.2 L'aggiunta di un controllo ActiveX a un progetto Visual Basic

Parte VIII

File di database e stampa

"QUESTO PROBLEMA DOVRÀ ESSERE RISOLTO NELLA VERSIONE 2"

In questa parte...

Pochi sono coloro che si preoccupano di imparare a utilizzare un programma di database e un numero ancora minore si interessa ai diversi formati di file che possono contenere i dati. Per soddisfare il maggior numero di persone, Visual Basic è in grado di leggere, scrivere e modificare le informazioni memorizzate in una vasta gamma di formati di file di database, tra cui Microsoft Access, dBASE, Paradox e FoxPro.

Oltre a garantire un'occupazione a centinaia di autori ed editori di libri di informatica, lo scopo della maggior parte dei programmi è quello di memorizzare informazioni e poi stamparle o visualizzarle di nuovo. Dopotutto, che utilità avrebbe un database se foste costretti a digitare nomi e indirizzi nel vostro computer ogni volta che lo riaccendete? In questa parte apprenderete tutto sulla creazione, la memorizzazione e la stampa delle informazioni per i vostri programmi Visual Basic.

Capitolo 32

Creare i file di database

In questo capitolo

- Rispondere alla domanda: "Che cosa diavolo è un database?"
- Organizzare un database
- Creare un database personalizzato

Nel caso abbiate bisogno di scrivere un programma che memorizza e recupera informazioni organizzate quali nomi e indirizzi oppure numeri di matricola nel caso di inventari o ancora numeri telefonici dei clienti, avete due scelte. Si può scrivere un programma Visual Basic con una grande quantità di codice BASIC per memorizzare, recuperare e organizzare le informazioni su disco fisso oppure si può prendere la strada più facile e utilizzare un file di database.

Utilizzando i file di database con Visual Basic, potete progettare tranquillamente la vostra interfaccia utente senza dovervi preoccupare di come salvare, recuperare e ordinare i vostri dati.

Per creare un file di database, è possibile utilizzare uno dei due sistemi esistenti. Sistema numero uno: correte in un negozio e spendete migliaia di lire per acquistare un programma di database separato con un nome bizzarro come Microsoft Access, Paradox o FoxPro. Sistema numero due: potete utilizzare lo speciale programma aggiuntivo (add-in) di Visual Basic denominato Visual Data Manager. Dato che in questo libro si è scelto di fare le cose nel modo più facile, il presente capitolo affronta principalmente l'utilizzo del Visual Data Manager (dopotutto viene fornito gratuitamente con Visual Basic).

Che cosa diavolo è un database?

Fondamentalmente, un file di database è semplicemente un file speciale progettato per contenere informazioni organizzate. Proprio come un file di un programma di elaborazione testi contiene le frasi e i paragrafi, un file di database contiene informazioni quali nomi, indirizzi, numeri telefonici, numeri delle tessere della mutua, numeri di scarpe, password segrete da vendere ad agenzie straniere e così via.

In genere un file di database è composto dai seguenti elementi, come illustrato nella Figura 32.1.

- Una o più tabelle
- Record
- Campi

Il *database* è solitamente un unico file che contiene tutte le informazioni.

Le *tabelle* permettono di organizzare le informazioni memorizzate all'interno di un database. Per esempio, anziché buttare nomi e indirizzi in un database come gettate i vestiti in un armadio, le tabelle vi permettono di memorizzare una serie di nomi e di indirizzi come Clienti e un'altra serie di nomi e di indirizzi come Fornitori, Destinatari posta, Scadenze e qualunque altro tipo di classificazione siate in grado di elaborare. Ogni database deve avere perlomeno una tabella.

Ogni tabella è composta da uno o più *record*, ognuno dei quali contiene tutte le informazioni relative a una singola voce, quali nome, indirizzo e numero telefonico di una persona. Nel mondo reale, il biglietto da visita di una persona è un record.

Ogni record è composto da uno o più *campi*; un campo contiene una singola parte di informazioni quali il nome, l'indirizzo, le dimensioni del cappello o lo stato di famiglia.

Figura 32.1 Gli elementi di un database

Organizzare la struttura di un database

Ora che ci si è fatti un'idea generale dello scopo di ogni parte di un file di database, bisogna decidere esattamente come organizzare le informazioni. Prima di creare un database è bene porsi questa domanda: "Di quale tipo di informazioni avrò bisogno nel futuro?".

Se per esempio si sta creando un database dei vostri contatti commerciali, potrebbe essere necessario memorizzare nomi, numeri telefonici, indirizzi di posta elettronica e numeri di fax. Se si sta creando un elenco per un inventario, si potrebbe voler memorizzare i numeri di matricola, le descrizioni dei pezzi, le quantità in magazzino e il prezzo per articolo.

Dopo che si sarà deciso il tipo di informazioni da salvare, si passerà a creare i nomi dei campi per specificare ogni informazione. I campi hanno due scopi.

- ✓ Memorizzare una parte di informazioni.
- ✓ Fornire un sistema per ordinare un database ed effettuare ricerche in esso.

Se volete, per esempio, memorizzare il nome "Gianni Bronco", potete memorizzarlo utilizzando uno dei seguenti metodi.

Metodo	*Nome campo*	*Dati effettivi*
n.1	Nome	Gianni Bronco
n.2	Nome di battesimo	Gianni
	Cognome	Bronco

Il primo metodo memorizza il nome della persona come un blocco completo di informazioni e questo significa che non è possibile eseguire una ricerca sul database per nome di battesimo o cognome.

Il secondo metodo memorizza il nome della persona in due blocchi di informazione; questo sistema può essere più noioso ma consente di eseguire ricerche sul database per nome di battesimo o per cognome.

Nessuno di questi è giusto o sbagliato: tutto dipende da come volete utilizzare i dati. Dopo avere stilato un elenco di tutti i campi in cui volete memorizzare le informazioni, dovrete decidere se volete suddividere il database in tabelle (categorie). Per esempio, se state inserendo i biglietti da visita, potreste volerli organizzare sotto forma di Contatti personali, Contatti commerciali, Contatti che mi devono un favore e così via. Nella Figura 32.2 viene illustrata una tipica progettazione di un database.

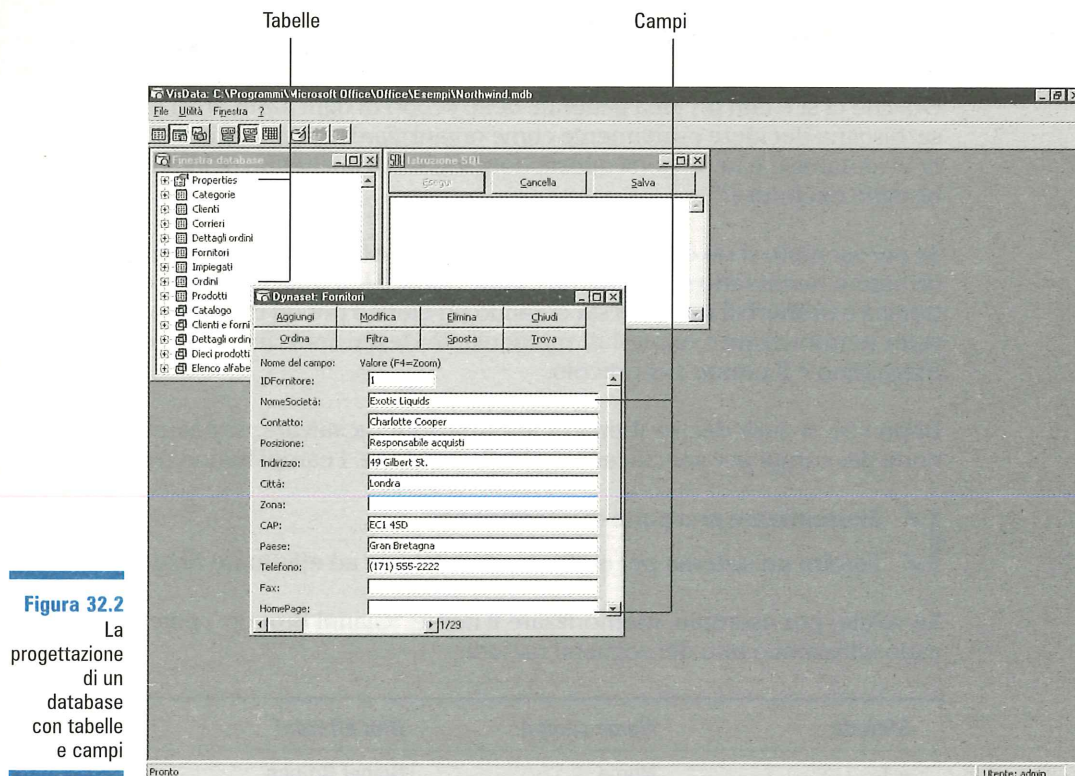



Figura 32.2 La progettazione di un database con tabelle e campi

Non si è costretti a dividere un database in due o più tabelle; infatti un database può anche essere composto da un'unica tabella.

Creare un file di database

Dopo che sarà stata progettata sulla carta la struttura del database, si passerà alla progettazione sul computer. Ricordate che, oltre a poter utilizzare un programma di database separato (come Microsoft Access o dBase), per svolgere questo compito è possibile anche usufruire del programma aggiuntivo di Visual Basic denominato Visual Data Manager.

Eseguire il programma Visual Data Manager

Per eseguire il programma Visual Data Manager, utilizzate la procedura illustrata qui di seguito.

1. **Caricate Visual Basic.**

2. **Selezionate Aggiunte-Visual Data Manager.**

   Appare la finestra VisData.

3. **Selezionate File-Nuovo.**

   Appare un elenco dei formati dei file di database, come Microsoft Access, Paradox e FoxPro.

4. **Selezionate Microsoft Access-MDB Versione 7.0.**

   Sarebbe meglio utilizzare un database Microsoft Access 7.0, a meno che dobbiate creare i file di database in un altro formato perché tutti quelli che conoscete utilizzano i file di database Paradox o FoxPro. Appare la finestra di dialogo Seleziona database Microsoft Access da creare.

5. **Digitate il nome del vostro file di database e poi fate clic su Salva.**

   Il Visual Data Manager visualizza una finestra di database.

6. **Posizionate il cursore del mouse sull'icona Proprietà nella Finestra database e fate clic con il pulsante destro.**

   Appare un menu di scelta rapida.

7. **Fate clic su Nuova tabella.**

   Appare la finestra di dialogo Struttura della tabella.

8. **Digitate il nome della tabella di database nella casella di testo Nome tabella.**

9. **Fate clic sul pulsante Aggiungi campo.**

   Appare la finestra di dialogo Aggiungi campo.

10. **Digitate il nome del campo nella casella Nome, per esempio Cognome o Età.**

11. **Fate clic sulla casella di riepilogo Tipo e selezionate il tipo di dati che volete memorizzare nel campo, come Testo, Valuta o Data/Ora, e digitate le dimensioni dei dati; poi fate clic su OK.**

12. **Ripetete i passi 10 e 11 per ogni campo che volete aggiungere. Al termine, fate clic su Chiudi.**

13. **Fate clic sul pulsante Crea tabella.**

   La Finestra database visualizza la tabella sotto forma di icona.

Dopo avere creato una tabella, potete sempre modificarla in un momento successivo. A questo scopo è sufficiente posizionare il cursore del mouse sull'icona della tabella nella Finestra database, fare clic con il pulsante destro e selezionare Progettazione.

Inserire le informazioni nel database

Dopo che si sono creati un file di database, una o più tabelle e uno o più campi per contenere le informazioni, vediamo come riempire il database di informazioni quali nomi di persone, numeri di telefono e così via.

Non siete costretti a inserire i dati in un database; potete anche lasciarlo vuoto in modo che qualcun altro che utilizza il vostro programma Visual Basic possa immettere le proprie informazioni nel database.

Per memorizzare informazioni nel database utilizzando il programma Visual Data Manager, assicuratevi che il programma Visual Data Manager sia stato avviato e procedete come illustrato di seguito.

1. **Nella Finestra database posizionate il cursore sull'icona corrispondente alla tabella, fate clic con il pulsante destro e selezionate Apri.**

   Appare una finestra di dialogo.

2. **Fate clic su Aggiungi.**
3. **Digitate le informazioni che volete memorizzare in ciascun campo.**
4. **Fate clic su Aggiorna quando avete terminato.**
5. **Fate clic su Chiudi.**

Uscire da Visual Data Manager

Per uscire dal programma Visual Data Manager, è sufficiente selezionare File-Esci.

Visual Data Manager è un programma semplice per la creazione di database. Se si ha la necessità di creare database complessi, sarebbe meglio acquistare una copia separata di Microsoft Access e utilizzarlo per creare i file di database.

Capitolo 33

Utilizzare i file dei programmi di database che non avete mai usato

In questo capitolo

- Collegarsi a un file di database
- Visualizzare record diversi
- Cercare record specifici

Se prevedete di memorizzare grandi quantità di dati, potreste utilizzare un file di database; Visual Basic è in grado di registrare e recuperare dati in uno qualsiasi dei quattro seguenti formati di database.

- ✓ File MDB di Microsoft Access
- ✓ File DBF di Borland dBASE
- ✓ File DBF di Microsoft FoxPro
- ✓ File DB di Borland Paradox

Se non avete la più pallida idea di cosa siano questi file o database, consideratevi fortunati.

Visual Basic è dotato di un programma separato denominato Visual Data Manager (per ulteriori informazioni leggete il Capitolo 32), che permette di creare file di database (MDB) Microsoft Access senza dover sborsare denaro per acquistare una copia di Microsoft Access. Tuttavia, per creare dei database veramente sofisticati, sarebbe meglio acquistare una copia separata del programma di database che volete utilizzare, come dBASE, Microsoft Access o Paradox.

Che cosa sono i file di database?

Ogni volta che uno di quegli splendidi programmi di database come Paradox, Access o dBASE salva informazioni, il programma memorizza queste informazioni in

un file su disco. Dato che il file contiene una serie di dati ritenuti importanti, viene chiamato *file di database*.

Per organizzare i dati, un file di database comprende uno o più record. Un record assomiglia a una scheda: ogni record contiene dei campi che contengono informazioni specifiche (nomi, indirizzi e cose simili). Nella Figura 33.1 viene riportato l'esempio di un record.

Figura 33.1 Un tipico record di database

Per organizzare ulteriormente queste informazioni, i file di database vi permettono di raggruppare i dati in tabelle. Una tabella è un sottoinsieme del database e comprende soltanto informazioni specifiche, come i nomi di tutte le persone che vivono nel Madagascar oppure i numeri telefonici di tutti quelli che vi stanno antipatici.

Collegarsi a un file di database

Affinché Visual Basic legga o modifichi un file di database, è necessario specificare quanto segue.

- ✓ Quale file di database utilizzare
- ✓ Quale tipo di record utilizzare
- ✓ Quale tabella di database utilizzare
- ✓ Quali campi di database visualizzare

Collegarsi manualmente a un database

Per collegare il programma a un file di database, si utilizzi la procedura di seguito riportata.

1. **Controllate che il file di database desiderato esista.**

   In caso contrario, dovete creare il file utilizzando un programma di database (come Paradox o FoxPro) oppure il programma Visual Data Manager.

2. **Fate clic sull'icona Data nella casella degli strumenti di Visual Basic e disegnate il controllo dei dati sulla form.**

   La Figura 33.2 mostra l'icona Data sulla barra degli strumenti e tracciata su una form. Per ulteriori informazioni su come utilizzare il controllo dati, leggete il paragrafo "Manipolare record diversi" più avanti in questo stesso capitolo.

3. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure fate clic sull'icona Finestra Proprietà sulla barra degli strumenti.**

4. **Fate doppio clic sulla proprietà DatabaseName.**

   Visual Basic visualizza la finestra di dialogo Nome database.

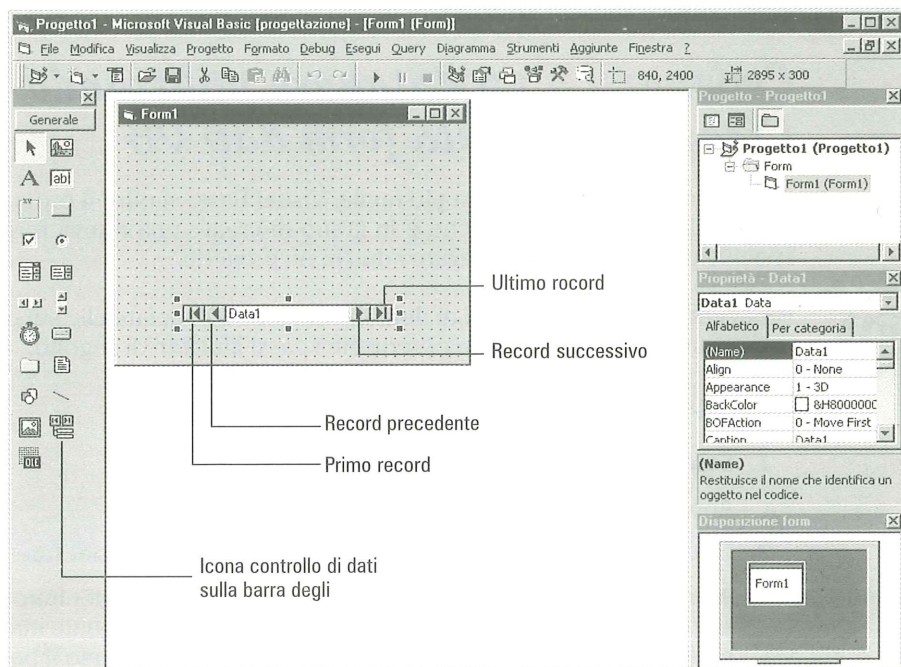



Figura 33.2 L'icona Data e il controllo dati disegnato su una form

5. **Selezionate il file di database che volete utilizzare, per esempio BIBLIO.MDB, il database d'esempio fornito con Visual Basic, e poi fate clic su Apri.**

6. **Fate clic sulla proprietà Recordset Type e poi fate clic sul pulsante grigio con la freccia nella casella delle impostazioni.**

   Visual Basic visualizza un elenco dei tipi di record tra cui scegliere:

   0 (Table)

   1 (Dynaset)

   2 (Snapshot)

7. **Fate clic sul tipo di record, come 0 (Table).**

8. **Fate clic sulla proprietà RecordSource e poi fate clic sul pulsante grigio con la freccia nella casella delle impostazioni.**

   Visual Basic visualizza un elenco di tabelle di database tra cui scegliere.

9 **Fate clic su una tabella di database (per esempio Titles, nel caso abbiate scelto il database BIBLIO in dotazione a Visual Basic).**

Se state utilizzando dei file MDB di Microsoft Access, selezionate il tipo di record Table nella fase 7. Se state utilizzando un altro tipo di database, selezionate 1- il tipo di record Dynaset. Se dovete semplicemente leggere i dati senza aggiornarli, tirate diritto e selezionate 2- il tipo di record Snapshot.

Collegarsi a un database utilizzando Creazione guidata form dati VB

Se volete imboccare la strada più facile, Visual Basic vi offre il programma Creazione guidata form dati VB, che è in grado di creare una nuova form progettata specificatamente per visualizzare i dati in un progetto.

Per accettare le impostazioni di default per la form dati, fate clic sul pulsante Fine in qualsiasi momento mentre percorrete i passi illustrati di seguito.

Ecco la procedura per utilizzare Creazione guidata form dati.

1. **Selezionate Progetto-Inserisci form.**

   Appare la finestra di dialogo Inserisci form.

2. **Fate clic sull'icona Creazione guidata form dati VB e poi fate clic su Apri.**

   Appare la finestra di dialogo Creazione guidata form dati - Introduzione, illustrata nella Figura 33.3. Se avete creato una form sulla quale intendete basare la vostra form attuale, fate clic sulla freccia rivolta verso il basso nella casella di riepilogo per selezionare un profilo da utilizzare.

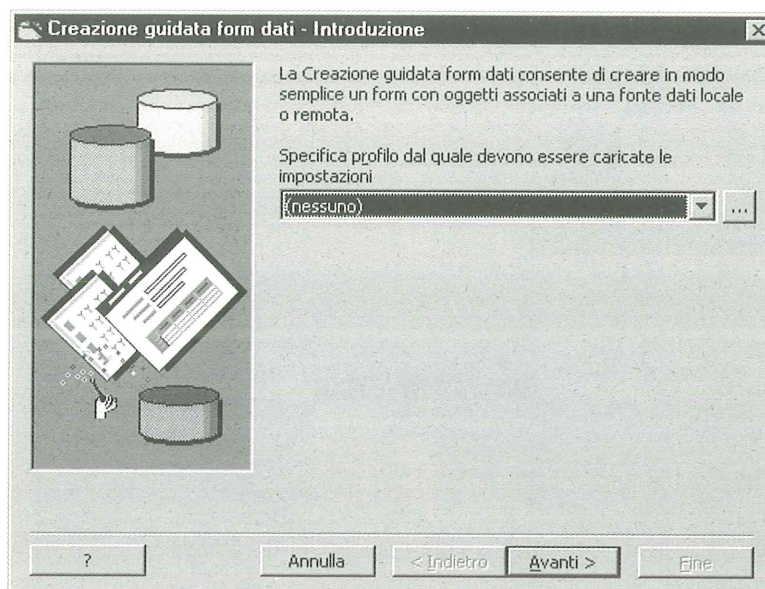



Figura 33.3 La finestra di dialogo Creazione guidata form dati - Introduzione

3. **Fate clic su Avanti.**

   Appare la finestra di dialogo Creazione guidata form dati - Tipo database.

4. **Fate clic sul formato di database prescelto (per esempio Access) e poi fate clic su Avanti.**

   Appare la finestra di dialogo Creazione guidata form dati - Database.

5. **Fate clic sul pulsante Sfoglia.**

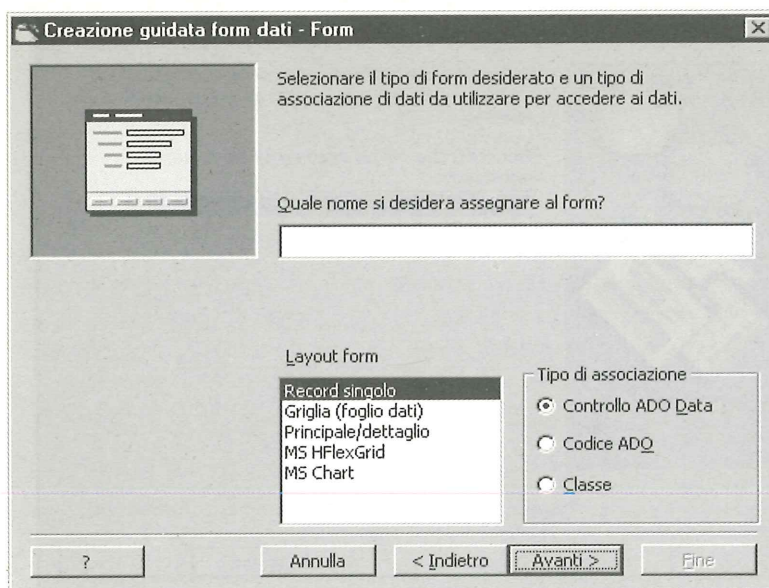
   Appare la finestra di dialogo Database Access.

6. **Fate clic sul file di database che volete utilizzare, fate clic su Apri e poi fate clic su Avanti.**

   Appare la finestra di dialogo Creazione guidata form dati - Form illustrata nella Figura 33.4.

7. **Digitate un nome per la form e fate clic su uno di questi pulsanti.**

   - **Record singolo:** crea una form che visualizza un record alla volta.
   - **Griglia (foglio dati):** crea una form che visualizza i dati nel formato di un foglio elettronico.
   - **Principale/dettaglio:** crea un record master che visualizza i dati sotto forma di un unico record e una sorgente di record di dettaglio sotto forma di foglio elettronico.

Figura 33.4 La finestra di dialogo Creazione guidata form dati - Form

- **MS HFlexGrid:** crea una form che visualizza i dati in formato foglio elettronico (griglia) dove i dati vengono fusi e consolidati mediante righe e colonne.
- **MS Chart:** crea una form che visualizza i dati sotto forma di grafico.

8. **Fate clic su Avanti.**

   Appare la finestra di dialogo Creazione guidata form dati - Origine record, illustrata nella Figura 33.5.

9. **Fate clic sulla freccia verso il basso nella casella di riepilogo Origine record e selezionate il record che volete visualizzare.**

10. **Fate clic su un campo visualizzato nella casella di riepilogo Campi disponibili e fate clic sul pulsante > per ogni campo che volete visualizzare sulla form; al termine, fate clic su Avanti.**

    Appare la finestra di dialogo Creazione guidata form dati - Seleziona controllo, illustrata nella Figura 33.6.

11. **Fate clic sulle caselle di controllo per ciascun tipo di pulsante che intendete visualizzare sulla form (come un pulsante Inserisci o Elimina), quindi fate clic su Avanti>.**

    Appare la finestra di dialogo Creazione guidata formati - Operazione completata, illustrata nella Figura 33.7.

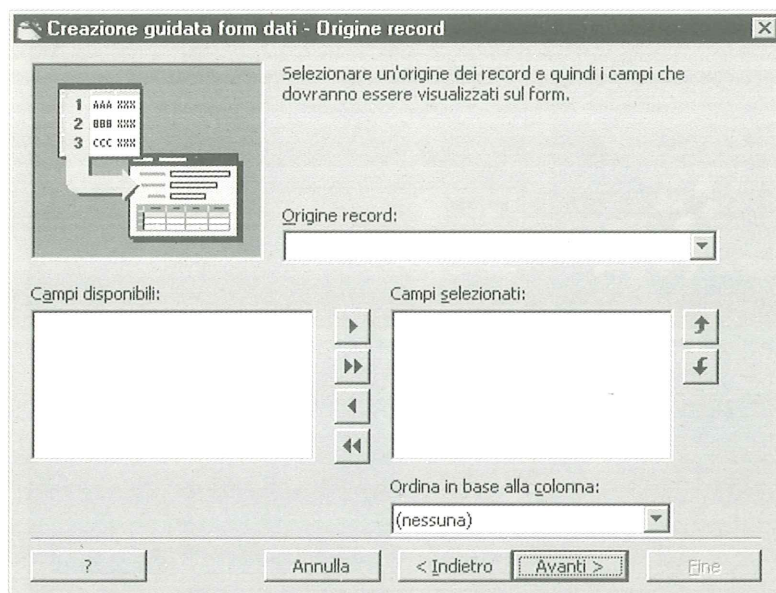



Figura 33.5 La finestra di dialogo Creazione guidata form dati - Origine record

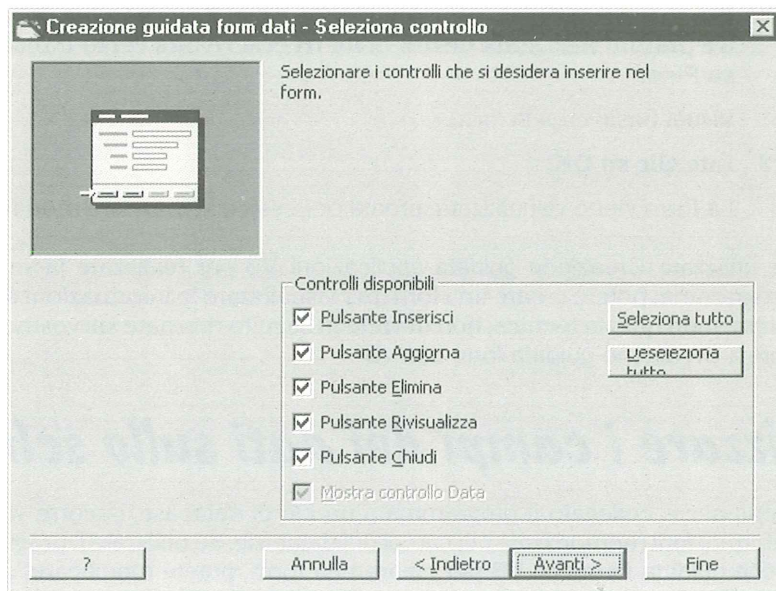
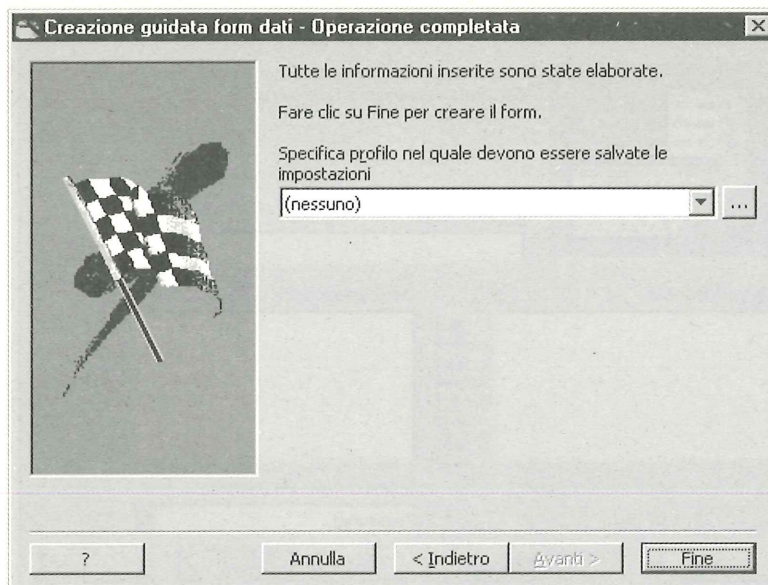



Figura 33.6 La finestra di dialogo Creazione guidata form dati - Seleziona controllo

Figura 33.7 La finestra di dialogo Creazione guidata form dati - Operazione completata

12. **Per salvare la form come un profilo da utilizzare in futuro, fate clic sui tre puntini neri sulla destra della freccia rivolta verso il basso. Fate clic su Fine.**

    Visual Basic crea la form.

13. **Fate clic su OK.**

    La form viene visualizzata, pronta per essere elaborata e modificata.

Se utilizzate Creazione guidata applicazioni VB per realizzare la struttura di un programma, potete creare una form per visualizzare le informazioni del database. Utilizzando questa tecnica, non dovrete in seguito ritornare sui vostri passi e ricorrere a Creazione guidata form dati VB.

Visualizzare i campi dei dati sullo schermo

Dopo avere collegato il programma a un file di database, occorre visualizzare le informazioni memorizzate nei campi del database. Se utilizzate il programma Creazione guidata form dati VB per creare una form, potete modificare, aggiungere o cancellare campi aggiuntivi dalla form. Visual Basic offre diversi metodi per visualizzare i campi di un database.

- ✓ Caselle di controllo

- ✓ Riquadri di disegno e di immagine
- ✓ Etichette e caselle testo
- ✓ Caselle di riepilogo e caselle combinate
- ✓ Griglie

Le caselle di controllo visualizzano valori Sì/No oppure True/False; se il campo contiene un valore Sì oppure True, la casella di testo appare selezionata sullo schermo. Se il campo contiene un valore No oppure False, la casella di testo appare vuota.

I riquadri di immagine e di disegno permettono di visualizzare immagini grafiche, purché il file di database contenga tali immagini.

Le caselle di testo e le etichette visualizzano informazioni quali nomi, indirizzi, numeri telefonici e quantità.

Le caselle di riepilogo e le caselle combinate permettono di visualizzare più scelte; le caselle combinate offrono all'utente la possibilità di digitare qualcosa che non compare nell'elenco.

Se volete offrire all'utente la possibilità di modificare i dati visualizzati, utilizzate una casella di testo; se invece volete evitare che l'utente non modifichi i dati, utilizzate un'etichetta.

Per creare una casella di testo, un'etichetta, una casella di controllo, un riquadro di immagine, una casella combinata, una casella di riepilogo, una griglia o un riquadro di immagine per visualizzare un campo di database, assicuratevi di disegnare almeno un controllo dati sulla form, quindi utilizzate la procedura di seguito riportata.

1. **Disegnate la casella di testo, l'etichetta, la casella di controllo, il riquadro di immagine, la casella combinata, la casella di riepilogo, la griglia oppure il riquadro di disegno sulla form che contiene il controllo dei dati.**
2. **Aprite la finestra Proprietà. A questo scopo, premete F4, selezionate Visualizza-Finestra Proprietà oppure fate clic sull'icona Finestra Proprietà sulla barra degli strumenti.**
3. **Fate clic sulla proprietà DataSource e poi fate clic sul pulsante grigio con la freccia nella casella delle impostazioni.**

   Visual Basic visualizza un elenco di tutti i nomi di controlli dei dati presenti sulla form.
4. **Selezionate un nome di controllo dei dati (come Data1).**

5. **Fate clic sulla proprietà DataField e poi fate clic sul pulsante grigio con la freccia nella casella delle impostazioni.**

   Visual Basic visualizza un elenco di tutti i nomi dei campi che potete visualizzare (saltate questa fase se avete disegnato una griglia nella fase 1).

6. **Fate clic sul nome di un campo.**

Quando selezionate un nome di campo per una casella di controllo, controllate che il campo di database contenga soltanto valori Sì/No oppure True/False. Quando selezionate un nome di campo per un riquadro di disegno o di immagine, verificate che il campo di database contenga solo immagini grafiche. In caso contrario, Visual Basic si confonde e non sa come gestire questi dati quando viene eseguito il programma.

Manipolare record diversi

Quando Visual Basic carica un database fa in modo che il primo record del database diventi quello attivo. Il comando che selezionate influirà solo su questo primo record.

Scegliere record diversi è un po' come consultare lo schedario di una biblioteca (i più giovani che non hanno mai visto uno schedario pensino agli espositori per CD nei negozi di articoli musicali).

Visual Basic offre due metodi per far in modo che un record diverso diventi il record attivo.

- ✓ Utilizzare l'oggetto controllo dei dati
- ✓ Utilizzare il codice BASIC

Utilizzare l'oggetto controllo dei dati per visualizzare i record in un database

L'oggetto controllo dei dati dispone di quattro frecce che permettono all'utente di scorrere i diversi record; nella Figura 33.2 sono illustrate queste frecce.

L'oggetto controllo dei dati è progettato per assomigliare ai dispositivi di controllo presenti su un mangianastri o su un videoregistratore; a proposito, quanti di voi sanno come programmare un videoregistratore?

La freccia sull'estrema sinistra e quella sull'estrema destra permettono di spostarsi al primo e all'ultimo record del database; le altre due frecce permettono di spostarsi in avanti o all'indietro procedendo di un record alla volta.

L'oggetto controllo dei dati consente all'utente di scegliere un record; verificate di avere visualizzato sullo schermo una casella di testo o un'etichetta in modo da mostrare i dati estratti dal record attivo. In caso contrario, l'utente non saprà mai qual è il record attivo e non sarà neppure in grado di visualizzare o modificare i dati.

Utilizzare il codice BASIC per visualizzare i record in un database

Per visualizzare il primo record mediante il codice BASIC, utilizzate il seguente comando:

```
Data1.Recordset.MoveFirst
```

In tutti questi comandi `Data1` si riferisce al nome dell'oggetto di controllo dei dati. Se modificate il nome di tale oggetto, sostituite `Data1` con il nome scelto da voi.

Per visualizzare l'ultimo record nel database, utilizzate il seguente comando:

```
Data1.Recordset.MoveLast
```

Per visualizzare il record successivo nel database, utilizzate il seguente comando:

```
Data1.Recordset.MoveNext
```

Per visualizzare il record precedente nel database, utilizzate il seguente comando:

```
Data1.Recordset.MovePrevious
```

Aggiungere un record

Per aggiungere un record a un database, utilizzate la procedura di seguito riportata.

1. **Create un nuovo record utilizzando il seguente codice:**

   ```
   Data1.Recordset.AddNew
   ```

2. **Inserite i dati nei campi di questo record appena creato utilizzando il seguente codice:**

   ```
   Data1.Recordset.Field ("NomeCampo") = "NuovoDato"
   ```

   NomeCampo indica il campo specifico che volete modificare all'interno del vostro database ed è necessario racchiudere il nome tra apici. NuovoDato rappresenta la nuova informazione che volete memorizzare nel campo e può essere una stringa o un numero. Se le informazioni sono rappresentate

da una stringa gli apici sono necessari, se invece si tratta di numeri potete anche non inserirle.

3. **Salvate il record nel database con il seguente codice:**

```
Data1.Recordset.Update
```

Visual Basic aggiunge sempre i nuovi record alla fine di un file di database.

Modificare un record

Per modificare un record esistente in un database, utilizzate la procedura di seguito riportata.

1. **Visualizzate il record che volete modificare.**
2. **Utilizzate il seguente comando per dire a Visual Basic che volete modificare questo record:**

```
Data1.Recordset.Edit
```

3. **Attribuite i nuovi dati al campo specifico del quale volete modificare il contenuto:**

```
Data1.Recordset.Fields ("NomeCampo") = "NuovoDato"
```

4. **Salvate il record nel database:**

```
Data1.Recordset.Update
```

Cancellare un record

Per cancellare un record da un database, visualizzate il record che volete eliminare e poi utilizzate il seguente comando BASIC:

```
Data1.Recordset.Delete
Data1.Recordset.MoveNext
```

Questo comando dice: "Vedi il record visualizzato in questo momento? Cancellalo e quindi visualizza il record successivo in modo che quello cancellato non sia più visibile".

Contare il numero dei record esistenti

Talvolta è utile sapere quanti record contiene un file di Visual Basic. Per calcolare questo valore, utilizzate il seguente comando:

```
Dim RecordTotali As Long
Data1.RecordSet.MoveLast
RecordTotali = Data1.Recordset.RecordCount
```

1. La prima riga dice: "Crea una variabile denominata `RecordTotali` e definiscila come variabile con dati di tipo `Long`".
2. La seconda riga dice: "Vai all'ultimo record presente nel database".
3. La terza riga dice: "Conta il numero di record e memorizza questo valore nella variabile `RecordTotali`".

Trovare un record specifico

Per trovare un record specifico in un database, dovete utilizzare uno dei seguenti comandi.

- ✓ `Data1.Recordset.FindFirst "criteri"`
- ✓ `Data1.Recordset.FindNext "criteri"`
- ✓ `Data1.Recordset.FindPrevious "criteri"`
- ✓ `Data1.Recordset.FindLast "criteri"`

I criteri di ricerca specificano il campo da ricercare e i dati che volete trovare. Per esempio, per trovare tutti i record con le informazioni sulle persone il cui cognome è Bianchi, il vostro criterio di ricerca sarà simile al seguente:

```
"Cognome = 'Bianchi'"
```

Notate che il dato oggetto della ricerca (`'Bianchi'`) è racchiuso tra virgolette singole; se le tralasciate, Visual Basic non capirà che cosa volete trovare e non farà un bel niente. Ovviamente, questo esempio presuppone che esista un campo denominato `Cognome`.

Se provate a ricercare i dati in un campo inesistente, verrà visualizzato un messaggio di errore.

Per trovare il primo record che soddisfi i vostri criteri di ricerca, utilizzate il seguente comando:

```
Data1.Recordset.FindFirst "Cognome = 'Bianchi'"
```

Dopo che avrete trovato un record che soddisfi i vostri criteri di ricerca, potrete cercarne altri. Per trovare il record successivo che soddisfi i vostri criteri di ricerca, utilizzate il seguente comando:

```
Data1.Recordset.FindNext "Cognome = 'Bianchi'"
```

Per trovare il record precedente che soddisfi i vostri criteri di ricerca, utilizzate il seguente comando:

```
Data1.Recordset.FindPrevious "Cognome = 'Bianchi'"
```

Per trovare l'ultimo record che soddisfi i vostri criteri di ricerca, utilizzate il seguente comando:

```
Data1.Recordset.FindLast "Cognome = 'Bianchi'"
```

Recuperare dei dati da un campo di un database

Prima di recuperare delle informazioni da un campo di un database, trovate il record che volete utilizzando i comandi `FindFirst`, `FindNext`, `FindLast`, `FindPrevious` e quindi prelevate le informazioni dal campo del database seguendo uno di questi due metodi.

- ✓ Create una casella di testo o un'etichetta e impostate la sua proprietà DataField sul campo desiderato.
- ✓ Utilizzate il codice BASIC per eliminare le informazioni dal database e attribuirle a una variabile.

```
Dim Memorizza As String
Memorizza = Data1.Recordset.Field("NomeCampo").Value
```

1. La prima riga dice: "Crea una variabile denominata `Memorizza` e definisci questa variabile come tipo di dati `String`".
2. La seconda riga dice: "Leggi i dati memorizzati nel campo `NomeCampo` e memorizza il valore nella variabile `Memorizza`".

Modificare i dati in un database

Per modificare il contenuto del campo di un database quando il campo viene visualizzato in una casella di testo, in una casella di controllo, in un riquadro di immagine o di disegno, è sufficiente digitare o incollare delle nuove informazioni.

Per modificare il contenuto di un campo di database mediante il codice BASIC, utilizzate i seguenti comandi:

```
Data1.Recordset.Edit
Data1.Recordset.Fields("NomeCampo").Value = NuovoValore
Data1.Recordset.Update
```

1. Il primo comando dice a Visual Basic: "Sto preparandomi a modificare dei dati in un database".
2. Il secondo comando sostituisce il valore corrente del campo `NomeCampo` con il valore memorizzato in `NuovoValore`.
3. Il terzo comando dice: "Salva le modifiche apportate al database".

Ogni volta che inserite, eliminate o modificate un record in un database utilizzando del codice BASIC, per aggiornare il file del database dovrete utilizzare il comando Update.

Mettete alla prova le vostre nuove conoscenze

1. **Quali sono le fasi che dovete seguire per fare in modo che il vostro programma in Visual Basic utilizzi un file di database?**

   a. Innanzitutto è necessario creare un file di database utilizzando un programma di database separato, come Access o Paradox. Quindi, occorre disegnare l'icona Data Control su una form, scegliere il nome del file di database per le proprietà DatabaseName e una tabella di database per la proprietà RecordSource dell'oggetto controllo dati.

   b. Acquistare un computer, riportarlo al negozio perché non funziona, assumere un programmatore specializzato in database, licenziare il programmatore specializzato in database e cedere alla disperazione.

   c. Visual Basic è davvero in grado di utilizzare i file di database? Ehi, dove posso trovare altre informazioni su questo argomento?

   d. Nessuno. È una domanda trabocchetto e mi rifiuto di palesare la mia ignoranza scegliendo una di queste risposte.

2. **Che cosa fanno queste quattro righe di codice BASIC?**

```
Data1.Recordset.AddNew

Data1.Recordset.Fields
("Domanda") = "Hai mai
accettato soldi da governi
stranieri?"

Data1.Recordset.Fields
("Risposta") = "Sì, ma ho
preteso che mi pagassero in
lire italiane."

Data1.Recordset.Update
```

   a. La prima riga cancella qualsiasi database presente sul vostro disco fisso, la seconda e la terza riga sono superflue e la quarta riga distrugge il resto dei file presenti sul disco fisso.

   b. La prima riga aggiunge un nuovo record al file di database; la seconda riga memorizza la stringa `"Hai mai accettato soldi da governi stranieri?"` nel campo di database denominato `"Domanda"`; la terza riga memo-

(continua)

rizza la stringa `"Sì, ma ho preteso che mi pagassero in lire italiane."` nel campo di database denominato `"Risposta"` e la quarta riga aggiorna il database con queste nuove informazioni e aggiunge il record alla fine del file di database.

c. La prima riga confonde Visual Basic, la seconda pone a Visual Basic una domanda, la terza visualizza la risposta di Visual Basic fornita dietro consiglio di un avvocato mentre la quarta ignora l'intero problema e spera che tutti dimentichino in fretta ciò che è accaduto.

d. Ehi! Stai prendendoti gioco di qualche pezzo grosso, vero?

Inserire un segnalibro su un record specifico

Potete impostare un segnalibro su un record in modo da poter recuperare rapidamente quest'ultimo in qualsiasi momento. Per creare un segnalibro, dovete creare una variabile e definirla come tipo di dati Variant o String. Per esempio:

```
Dim Segnalibro1 As Variant
```

oppure

```
Dim Segnalibro2 As String
```

Ogni segnalibro può individuare un solo record.

Per inserire un segnalibro, visualizzate il record che volete contrassegnare e quindi utilizzate il seguente comando:

```
Segnalibro1 = Data1.Recordset.Bookmark
```

Per tornare a un segnalibro, utilizzate il seguente comando:

```
Data1.Recordset.Bookmark = Segnalibro1
```

Nota. Non tutti i campi di un database permettono di utilizzare i segnalibri; i file MDB di Microsoft Access sono sempre pronti a ospitarli, ma altri tipi di database potrebbero non esserlo; la disponibilità dipende dall'esistenza di un indice relativo a quel particolare database (se non sapete che cos'è un indice, non preoccupatevi e ignorate tranquillamente tutta questa discussione sugli indici). Per determinare se un file di database supporta i segnalibri, esaminate la proprietà Bookmarkable, come nell'esempio seguente:

```
If Data1.Recordset.Bookmarkable = True Then
   ' È possibile impostare dei segnalibri
```

```
ElseIf Data1.Recordset.Bookmarkable = False Then
  ' Non è possibile impostare dei segnalibri
End If
```

Fate attenzione ai formati di database; ogni volta che una società come Microsoft aggiorna i propri programmi di database (per esempio Access), la nuova versione del programma potrebbe utilizzare un formato di file diverso. Questo significa che Visual Basic potrebbe non essere in grado di utilizzare il nuovo formato di file aggiornato.

Capitolo 34

Come fare in modo che il vostro programma stampi

In questo capitolo

- Stampare il vostro progetto Visual Basic
- Stampare una form
- Utilizzare l'oggetto stampante per stampare linee e cerchi
- Contare le pagine

Un programma può acquisire i dati, manipolarli e visualizzarli sullo schermo ma, alla fine, potrebbe anche dover stampare qualcosa. Visual Basic offre due metodi per stampare.

- ✓ Visualizzare i dati sulla form e quindi stamparla.
- ✓ Inviare i dati direttamente alla stampante utilizzando l'oggetto Printer.

Stampare una form

Stampare una form è il metodo più semplice per stampare informazioni. Per stampare una form, utilizzate la seguente sintassi:

```
NomeForm.PrintForm
```

Questo comando dice a Visual Basic: "Trova la form denominata `NomeForm` e inviala alla stampante".

Per stampare una form denominata `frmInfo`, utilizzate il seguente comando:

```
frmInfo.PrintForm
```

È possibile stampare qualsiasi form del vostro programma, comprese le form invisibili e le form ridotte a icona. L'unico svantaggio della stampa delle form è che la qualità della risoluzione di stampa può non essere elevata.

Quasi tutti gli schermi utilizzano una risoluzione di 96 punti per pollice; (dpi) la maggior parte delle stampanti laser può stampare con risoluzione di 300 o 600 punti per pollice. Per una stampa rapida, il comando PrintForm funziona benissimo, ma per stampare con una risoluzione più elevata sarebbe meglio utilizzare l'oggetto Printer.

Stampare con l'oggetto Printer

L'oggetto Printer rappresenta uno spazio di memorizzazione temporanea che intercetta i dati, li ripulisce e quindi li invia alla stampante. Per stampare una form utilizzando l'oggetto Printer, bisogna in pratica ridisegnare la form sull'oggetto Printer utilizzando il codice BASIC.

Il vantaggio dell'oggetto Printer è che questo oggetto utilizza la risoluzione della vostra stampante; lo svantaggio è che dovete scrivere un codice BASIC consistente solo per stampare una semplice form.

Prima di stampare qualcosa utilizzando l'oggetto Printer, dovete definire i margini superiore/inferiore e destra/sinistra.

Per definire il margine superiore della pagina, utilizzate la seguente sintassi, dove `Superiore` rappresenta il valore del margine superiore:

```
Printer.ScaleTop = Superiore
```

Per definire il margine inferiore della pagina, utilizzate la seguente sintassi, dove `Inferiore` rappresenta il valore del margine inferiore:

```
Printer.ScaleBottom = Inferiore
```

Per definire il margine sinistro della pagina, utilizzate la seguente sintassi, dove `Sinistra` rappresenta il valore del margine sinistro:

```
Printer.ScaleLeft = Sinistra
```

Per definire il margine destro della pagina, utilizzate la seguente sintassi, dove `Destra` rappresenta il valore del margine destro:

```
Printer.ScaleRight = Destra
```

Potrebbe essere necessario sperimentare diversi valori prima di trovare quello giusto.

Stampare un testo sull'oggetto Printer

Prima di poter stampare un testo mediante l'oggetto Printer, è necessario definire la posizione X e Y nella quale si vuole far apparire il testo.

Per definire questa posizione, utilizzate la seguente sintassi:

```
Printer.CurrentX = ValoreX
Printer.CurrentY = ValoreY
```

Questi comandi dicono: "Comincia a stampare tutto il testo partendo dalla posizione definita dalle proprietà CurrentX e CurrentY".

Per cominciare a stampare partendo dall'angolo superiore sinistro della pagina, utilizzate i seguenti comandi:

```
Printer.CurrentX = 0
Printer.CurrentY = 0
```

Dopo che avrete definito dove cominciare la stampa, passate alla fase operativa. Per stampare un testo sull'oggetto Printer, utilizzate la seguente sintassi:

```
Printer.Print "Stringa di testo"
```

Questo comando dice: "Colloca una stringa di testo sull'oggetto Printer e stampala nella posizione definita in precedenza dalle proprietà CurrentX e CurrentY".

Stampare linee e cerchi sull'oggetto Printer

La stampa di un normalissimo testo può essere noiosa e, per questo motivo, Visual Basic offre l'opportunità di vivacizzare i vostri capolavori utilizzando linee e cerchi. Se tutto ciò vi dà l'impressione che Visual Basic stia trasformando il vostro super computer e la vostra splendida stampante laser in un programmino di disegno, avete perfettamente ragione.

Definire lo spessore delle linee

Prima di cominciare a disegnare le linee e i cerchi, dovete definirne lo spessore: minore è lo spessore, più le vostre linee risulteranno sottili.

Per definire lo spessore, utilizzate la seguente sintassi:

```
Printer DrawWidth = Valore
```

Per definire la linea più sottile possibile, utilizzate questo comando:

```
Printer DrawWidth = 1
```

Mettete alla prova le vostre nuove conoscenze

1. **Perché mai il vostro programma Visual Basic dovrebbe stampare qualcosa?**

   a. Per sprecare carta e contribuire alla deforestazione globale dei paesi del Terzo Mondo.

   b. In modo che io possa giustificare l'acquisto della mia stampante laser da 3 milioni.

   c. Per fornire copie cartacee di tutte le informazioni importanti che il mio programma Visual Basic potrebbe creare.

   d. Non esiste alcuna ragione per stampare qualcosa. Osservando gli sviluppi della politica non avete ancora imparato che è sempre meglio non mettere mai nero su bianco qualcosa che potrebbe in seguito danneggiarvi?

2. **Perché quando si stampa un testo oppure si disegnano righe e cerchi è necessario specificare le coordinate X e Y?**

   a. Per avere l'impressione che, in fondo, il liceo è servito a qualcosa.

   b. In modo da dire a Visual Basic con esattezza in che punto della pagina volete far comparire le vostre linee, i vostri cerchi o il testo.

   c. Le coordinate x e y definiscono la posizione della vostra stampante rispetto al computer. Se sbagliate a definire le coordinate x e y, i vostri documenti potrebbero uscire dal tostapane o dal condizionatore.

Per definire uno spessore di linea più consistente, utilizzate questo comando:

```
Printer DrawWidth = 5
```

Per chi volesse proprio saperlo, i valori di `DrawWidth` possono variare da 1 a 32.767. 1 specifica una linea con lo spessore di un pixel; 32.767 specifica una linea con uno spessore di 32.767 pixel, vale a dire più di quanto avrete bisogno in tutta la vostra vita.

La stampa di una linea con lo spessore di un pixel su una stampante laser a 600 dpi potrebbe avere un aspetto diverso dalla stampa di una linea con lo spessore di un pixel su una stampante a getto d'inchiostro a 300 dpi.

Disegnare linee sull'oggetto Printer

Per disegnare una linea utilizzate la sintassi di seguito riportata, dove x1 e y1 definiscono il punto di partenza della linea, mentre x2 e y2 ne definiscono il punto di arrivo:

```
Printer.Line(x1,y1) - (x2,y2)
```

Disegnare cerchi sull'oggetto Printer

Per disegnare un cerchio utilizzate la sintassi di seguito riportata, dove x e y rappresentano il centro del cerchio, mentre Raggio definisce il raggio del cerchio:

```
Printer.Circle(x,y), Raggio
```

Stampare più pagine

Di solito Visual Basic continua a stampare i dati fino al termine della pagina, dopodiché si interrompe e comincia a stampare automaticamente su una pagina nuova.

Tuttavia, se volete che a un certo punto Visual Basic cominci a stampare su nuova pagina, utilizzate il seguente comando:

```
Printer.NewPage
```

Questo comando dice a Visual Basic di cominciare a stampare sulla pagina nuova (Fenomenale! Che cosa inventeranno ancora?).

Definire la qualità di stampa

In base alla stampante di cui si dispone, è possibile specificare una serie di risoluzioni per la stampa; il comando magico per controllare la risoluzione di stampa è:

```
Printer.PrintQuality = x
```

dove x rappresenta un numero negativo tra -4 e -1 oppure un numero positivo che indica la risoluzione specifica che intendete utilizzare, misurata in punti per pollice (dpi). Per esempio, nella seguente tabella viene riportata la risoluzione di stampa con numeri negativi da -4 a -1.

```
Printer.PrintQuality = -1   Risoluzione bozza
Printer.PrintQuality = -2   Risoluzione bassa
Printer.PrintQuality = -3   Risoluzione media
Printer.PrintQuality = -4   Risoluzione alta
```

Se preferite utilizzare le costanti predefinite di Visual Basic, potete servirvene nel seguente modo.

Costante	*Valore*	*Risoluzione di stampa*
vbPRPQDraft	-1	Risoluzione bozza

(continua)

Costante	*Valore*	*Risoluzione di stampa*
vbPRPQLow	-2	Risoluzione bassa
vbPRPQMedium	-3	Risoluzione media
vbPRPQHigh	-4	Risoluzione alta

Se preferite specificare la risoluzione esatta da utilizzare, inserite un numero positivo. Per definire 300 punti per pollice, vale a dire la risoluzione che è possibile ottenere con la maggior parte delle stampanti laser economiche, potete utilizzare il seguente comando:

```
Printer.PrintQuality = 300
```

Tenere conto delle pagine

Quando si stampano più pagine, Visual Basic tiene conto delle pagine in una proprietà denominata Page. Per utilizzare questo conteggio, servitevi del seguente comando BASIC:

```
Printer.Page
```

Quando avete finito di stampare

Per comunicare a Visual Basic che avete terminato di stampare, utilizzate il seguente comando:

```
Printer.EndDoc
```

Se dimenticate di utilizzare questo comando, indovinate che cosa succede? Visual Basic presuppone che il comando esista lo stesso, per cui inserirlo non è indispensabile. Tuttavia, è buona abitudine ricorrere a questo comando in modo da definire con chiarezza quando il programma ha finito di stampare.

Nel caso desideriate interrompere la stampa, potete dire a Visual Basic: "Ehi, ho cambiato idea. Smetti di stampare". Per interrompere immediatamente la stampa, utilizzate il seguente comando:

```
Printer.KillDoc
```

Come potete vedere, tentare di torturare Visual Basic per indurlo a stampare i dati può essere scomodo e noioso. Se avete tempo, scrivete le vostre routine di stampa e utilizzatele per tutti i vostri programmi. Altrimenti, quale alternativa più rapida, acquistate un controllo ActiveX chiamato vsView da ViewSoft (`www.videosoft`

.com). Questo controllo ActiveX consente di aggiungere con facilità capacità di stampa al vostro programma senza dover scrivere molto codice BASIC.

Provate anche voi

Il programma d'esempio di seguito riportato stampa un breve messaggio quando fate clic sul pulsante di comando Fare clic per stampare. Per creare questo programma, utilizzate le impostazioni specificate nella seguente tabella.

Oggetto	Proprietà	Valore
Form	Caption	Esempio stampa
Comando1	Caption	Fare clic per stampare
	Name	cmdStampa

Digitate il codice seguente nella finestra Codice. Se volete risparmiare uno sforzo eccessivo alle vostre dita, caricate il file PRINTME.VBP dal CD-ROM fornito insieme al libro.

```
Private Sub cmdStampa_Click()
Dim PagineTotali As Integer
Dim ContaPagine As String
  ' Specifica la posizione del testo da stampare
  Printer.CurrentX = 100
  Printer.CurrentY = 100
  Printer.Print "Questo appare in cima alla pagina"

  ' Specifica lo spessore e la posizione di una linea
  Printer.DrawWidth = 3
  Printer.Line(100, 100)-(10000,100)
  Printer.Line(100, 350)-(10000,350)
  PagineTotali = Printer.Page

  ' Specifica lo spessore e la posizione di una linea
  Printer.Circle (2000, 3500), 450
  Printer.CurrentX = 2750
  Printer.CurrentY = 3500
  Printer.Print "Questo è un cerchio."

  ' Specifica dove stampare il numero di pagina
  Printer.CurrentX = 1000
  Printer.CurrentY = 400
  ContaPagine = "PagineTotali = " & Str&(PagineTotali)
  Printer.Print ContaPagine
  Printer.EndDoc
End Sub
```

Stampare un progetto di Visual Basic

In questo capitolo è stato illustrato come fare in modo che il vostro programma Visual Basic stampi i dati mentre è in esecuzione. Tuttavia, potreste avere un'altra domanda: "Come diavolo faccio a stampare il codice sorgente nel mio progetto di Visual Basic?".

Fortunatamente, dopo tutto il tempo e gli sforzi che avete dedicato alla scrittura di un programma in Visual Basic, potete stamparlo per farlo ammirare ai posteri. Visual Basic offre tre metodi per stampare un progetto.

- ✓ **Form Image:** stampa la form proprio come appare sullo schermo.
- ✓ **Code:** stampa soltanto il codice BASIC.
- ✓ **Form As Text:** stampa i valori della proprietà di tutte le form e degli oggetti che compongono l'interfaccia utente.

Per stampare un progetto di Visual Basic, utilizzate la procedura di seguito riportata.

1. **Selezionate File-Stampa oppure premete CTRL+P.**

   Visual Basic visualizza la finestra di dialogo Stampa.

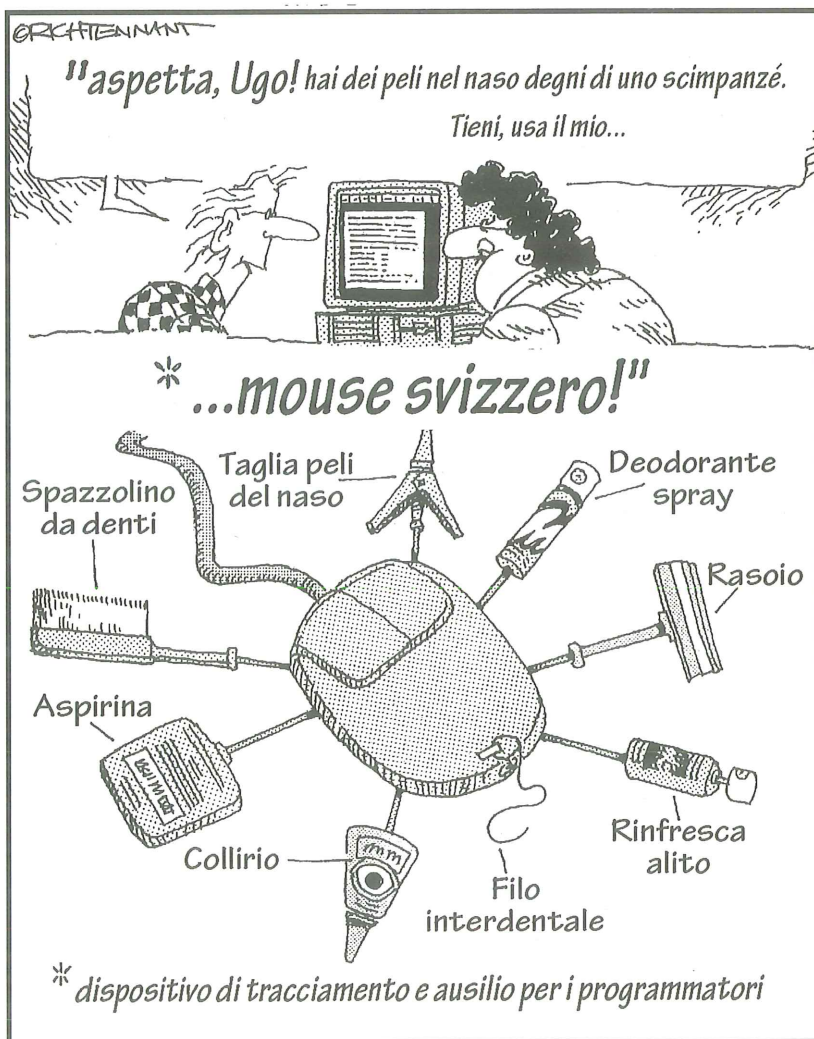
2. **Fate clic sul pulsante di opzione Modulo corrente per stampare soltanto il file evidenziato nella finestra Gestione progetti oppure sul pulsante di opzione Progetto corrente per stampare tutto il progetto di Visual Basic.**

3. **Selezionate una o più delle seguenti caselle di controllo.**

   - **Form.**
   - **Codice.**
   - **Testo form.**

4. **Controllate che la stampante sia accesa e fate clic su OK.**

Parte IX

La parte dei 10

In questa parte...

Ora che siete giunti fino a questo punto (se non siete quel tipo di lettori che saltano subito alla fine del libro), siete pronti per qualche idea che potrà aiutarvi a scrivere i migliori programmi possibili senza impazzire.

Con l'aiuto di questo libro e di Visual Basic, ora potete scrivere da soli il vostro programma o trovare un lavoro creando programmi per altri. Non è importante che tipo di programma dovete progettare, leggete comunque questa parte del libro per ulteriori suggerimenti sulla creazione dei vostri nuovi programmi Visual Basic.

Capitolo 35

Argomenti di Visual Basic che non c'entravano con nient'altro

In questo capitolo

- Trovare riviste e newsletter
- Visitare i siti Web Visual Basic
- Entrare a far parte di un user group
- Partecipare a conferenze tecniche
- Scrivere propri file DLL
- Scrivere programmi in Visual Basic per Macintosh e Linux

Ora che avete raggiunto la fine del libro (anche se siete passati direttamente a questa parte sfogliando il testo in libreria), forse vi starete chiedendo che cosa potete fare adesso per approfondire la vostra conoscenza di Visual Basic senza dover procedere per tentativi e rischiare di impazzire.

Di seguito è riportato un elenco di fonti nelle quali si possono trovare ulteriori informazioni su Visual Basic e ottenere l'aiuto necessario per creare un programma Visual Basic il più potente possibile.

Il Visual Basic Programmer's Journal

Ogni mese, cercate il nuovo numero del *Visual Basic Programmer's Journal* presso la vostra edicola o libreria. Questa rivista contiene numerosi articoli che analizzano gli intricati dettagli di Visual Basic, recensioni dei programmi aggiuntivi nonché esempi di codice che potete copiare (rubare) e utilizzare.

Contrariamente a quanto avviene nelle altre riviste, qui una delle presenze più utili è la pubblicità. Se avete bisogno di un controllo ActiveX per rendere più semplice la vostra attività di programmazione, scorrete gli annunci e probabilmente troverete ciò che fa per voi. Per ulteriori informazioni su questa rivista, rivolgetevi al seguente indirizzo.

Visual Basic Programmer's Journal

Fawcette Technical Publications
209 Hamilton Avenue
Palo Alto, CA 94301-2500
Tel: 415-833-7100
Fax: 415-853-0230
`www.windx.com`

Le newsletter Visual Basic

Per informazioni più dettagliate, acquistate una newsletter VB. Una società chiamata The Cobb Group pubblica newsletter mensili relative a diversi programmi, tra cui Paradox, Microsoft Office e Visual Basic. La newsletter *Inside Visual Basic* offre numerosi esempi di codice sorgente che potete studiare, modificare e rubare per fare colpo sul vostro capo.

Un'altra società, Pinnacle Publishing, pubblica una newsletter mensile Visual Basic Developer con allegato un dischetto che contiene esempi di codice sorgente. Naturalmente questa newsletter costa un po', ma se riuscite a convincere la vostra società ad acquistarla, che importanza ha il prezzo?

Per avere ulteriori informazioni su queste due pubblicazioni, si possono controllare i seguenti indirizzi.

Inside Visual Basic

The Cobb Group
9420 Bunsen Parkway
Louisville, KY 40220
Tel: 502-493-3300
Fax: 502-491-8050
`www.cobb.com`

Visual Basic Developer

Pinnacle Publishing, Inc
P.O. Box 888
Marietta, GA 30007-2255
Tel: 770-565-1763
Fax: 770-565-8232
`www.pinpub.com`

I siti Web dedicati a Visual Basic

Su Internet si possono trovare siti Web interamente dedicati alla programmazione in Visual Basic. Di seguito viene riportato un breve elenco dei siti Web più popolari, ma è bene ricordare che il numero di siti Web specializzati in Visual Basic continua ad aumentare.

Carl and Gary's Visual Basic Home Page:

`www.apexsc.com/vb`

Chris & Tim's VB Programming Resources al seguente indirizzo Web:

www.zetnet.co.uk/rad/index.html

VB Tips & Tricks Home Page:

www.vbtt.com

Partecipate a una conferenza tecnica Visual Basic

A distanza di pochi mesi Microsoft e il *Visual Basic Programmer's Journal* sponsorizzano seminari tecnici Visual Basic negli Stati Uniti, in Europa e in Asia. Queste conferenze sono occasioni eccezionali durante le quali è possibile apprendere tecniche da veri esperti di programmazione in Visual Basic, ascoltare le ultime notizie pubblicitarie dai rappresentati Microsoft, acquistare programmi aggiuntivi Visual Basic a prezzo ridotto dai fornitori e stabilire numerosi contatti con il mondo di Visual Basic. Per avere ulteriori informazioni su queste conferenze, contattate il seguente indirizzo.

Visual Basic Programmer's Journal

Fawcette Technical Publications
209 Hamilton Avenue
Palo Alto, CA 94301-2500
Tel: 415-833-7100
Fax: 415-853-0230
www.windx.com

Gli acquisti per corrispondenza

Non acquistate Visual Basic o qualunque altro programma aggiuntivo Visual Basic direttamente dall'editore. La maggior parte degli editori di prodotti software vi addebita il prezzo intero al dettaglio per i programmi.

Invece di acquistare direttamente dall'editore del software, acquistate piuttosto per corrispondenza. In questo caso, è probabile che riusciate ad avere sconti maggiori.

Due noti rivenditori per corrispondenza sono VBxtras e ZAC Catalog. Entrambe le società sono specializzate nella vendita di add-on Visual Basic con notevoli sconti, quindi chiamatele e chiedete un catalogo gratuito.

Per ulteriori informazioni su queste società di vendita per corrispondenza rivolgetevi al seguente indirizzo.

ZAC Catalog

1090 Kapp Drive
Clearwater, FL 33765-2111
Tel: 813-298-1181
Fax: 813-461-5808
`www.zaccatalog.com`

VBxtras

1905 Powers Ferry, Suite 100
Atlanta, GA 30339
Tel: 770-952-6356
Fax: 770-952-6388
`www.vbxtras.com`

Ampliate Visual Basic scrivendo i file DLL personalizzati

Se si prevede di scrivere diversi programmi Visual Basic, bisognerà organizzare le procedure utilizzate più di frequente in file modulo BAS separati. In questo modo si potranno inserire velocemente i file BAS in qualunque programma Visual Basic.

Purtroppo, i file BAS possono contenere soltanto i comandi Visual Basic. Dato che linguaggi come C++ e Pascal possono essere più flessibili quando si tratta di penetrare nelle viscere del vostro computer e manipolare singoli bit e byte, molti programmatori scrivono le procedure utilizzate più di frequente in C++ o in Pascal e le memorizzano in file DLL (Dynamic Link Library) separati. Un file DLL contiene le procedure più utilizzate che si possono condividere con programmi diversi.

Un file DLL può essere scritto utilizzando C++, Pascal o persino Visual Basic. Tuttavia, a prescindere dal linguaggio utilizzato, un file DLL permette al programma di funzionare più velocemente rispetto a quando ci si limita a memorizzare il codice BASIC in un file modulo BAS. (Per ulteriori informazioni su come scrivere file DLL, procuratevi l'ultima edizione di *MORE VB for Windows for Dummies).*

Sebbene possa essere difficile utilizzare i file DLL, risulta comunque più facile rispetto alla seconda alternativa che avete a disposizione, ovvero utilizzare l'API di Windows (si veda il paragrafo successivo).

Ampliate Visual Basic utilizzando l'API di Windows

Visual Basic rende così semplice la programmazione perché isola da tutti quei dettagli ingarbugliati necessari per scrivere i programmi Windows. Ma il prezzo

che si paga per questo isolamento è la mancanza di flessibilità che i programmatori in C++ amano ostentare (mentre diventano lentamente pazzi affrontando le complessità della programmazione in Windows).

Alla fine si troverà frustrante la facilità di utilizzo di Visual Basic, dato che non sarà possibile eseguire determinate attività. L'alternativa più veloce e più semplice è acquistare un programma aggiuntivo per dotare Visual Basic delle funzioni che si vogliono utilizzare. Ma se non si riesce a trovare un tale programma, bisognerà scavare nei meandri oscuri di Windows.

Per aiutare i programmatori a scrivere per Windows, Windows offre molti comandi denominati Microsoft Windows API (Application Programming Interface). Sebbene questi comandi siano complicati da imparare e capire, la loro complessità rende l'idea delle possibilità di gestire Windows a proprio piacimento.

Se amate vivere pericolosamente, trovate il tempo necessario per approfondire l'API di Windows (troverete maggiori dettagli sulle API di Windows nell'ultima edizione di *MORE VB for Dummies*) altrimenti fate finta che questo paragrafo non sia mai esistito e passate a un'altra parte del libro.

Acquistate un programma per creare i file della guida

Ogni programma Windows valido ha un sistema di guida online, in modo che l'utente in preda al panico possa sfogliare i riferimenti ipertestuali sullo schermo invece di procedere a stento in manuali di centinaia di pagine che non hanno comunque alcun senso. Se si è seriamente intenzionati a scrivere programmi Visual Basic, bisognerà fornire anche un sistema di guida insieme al programma.

Creare un sistema di guida non è difficile ma soltanto incredibilmente noioso. Fortunatamente ci sono programmi speciali che rendono più divertente questo processo. Due popolari programmi di creazione dei file della guida sono RoboHelp e ForeHelp, che permettono di progettare le schermate della guida con la stessa facilità con cui si scrive un documento in un programma di elaborazione testi. Una volta terminato, entrambi questi programmi vi permettono di testare i vostri capolavori mostrando con precisione l'aspetto che avranno le vostre schermate della guida quando verranno aggiunte al programma Visual Basic.

Nel mondo competitivo dello sviluppo di prodotti software, un valido file di guida è fondamentale per conferire un aspetto professionale al vostro programma. Anche in questo caso, se il vostro obiettivo non è quello di rendere più semplice l'utilizzo del programma, andate piuttosto a lavorare per una delle principali società di software.

Per avere ulteriori informazioni su RoboHelp e ForeHelp, contattate i seguenti indirizzi

RoboHelp

Blue Sky Software
7777 Fay Avenue, Suite 201
La Jolla, CA 92307
Tel: 619-459-6365
Fax: 619-459-6366
www.blue-sky.com

ForeHelp

ForeFront, Inc.
4710 Table Mesa Drive, Suite B
Boulder, CO 80303
Tel: 303-499-91981
Fax: 303-494-5446
www.ff.com

Acquistate un programma per creare i dischi di installazione

Dopo che sarà stato creato un programma in Visual Basic, si passerà alla sua distribuzione. Anche se è possibile copiare semplicemente il programma Visual Basic su dischetto e fare affidamento sul fatto che la persona che lo riceve sappia cosa fare, sarebbe meglio utilizzare invece un programma speciale di installazione.

I programmi di installazione possono guidare un utente, passo dopo passo, attraverso il processo spesso complicato di copia dei programmi su un altro computer. Visual Basic è dotato di un programma di installazione chiamato Application Setup Wizard, ma sarebbe comunque meglio acquistare un programma speciale di installazione, che offra la possibilità di personalizzare l'installazione con il logo della società, gli annunci pubblicitari e gli effetti sonori.

Due programmi popolari di installazione sono InstallShield e PC-Install. Entrambi questi programmi possono aiutarvi a creare programmi di installazione a prova di errore per tutti i vostri programmi Visual Basic.

Per avere ulteriori informazioni su questi programmi di installazione, contattate i seguenti indirizzi.

PC-Install

20/20 Software, Inc.
8196 SW Hall Blvd., Suite 200
Beaverton, OR 97008
Tel: 503-520-0504
Fax: 503-520-9118
www.twenty.com/~twenty

InstallShield

InstallShield Corporation
900 National Parkway, Suite 125
Schaumburg, IL 60173-5108
Tel: 847-240-9111
Fax: 847-240-9120
www.installshield.com

Scrivere programmi in Visual Basic per Macintosh e Linux

Quando Microsoft presentò Visual Basic nel lontano 1991, i programmatori se ne innamorarono subito perché consentiva loro di progettare applicazioni in maniera rapida e semplice. Per questa ragione, Microsoft promise che avrebbe adattato Visual Basic ad altri sistemi operativi in modo da consentire loro di scrivere programmi in Visual Basic per altri computer.

Dopo aver rilasciato una versione di Visual Basic per MS-DOS e averla lasciata morire, Microsoft si è sempre limitata ad accennare ad altre release future di Visual Basic per altre piattaforme. Quindi, se intendete utilizzare Visual Basic per scrivere un programma per Macintosh, non potete farlo.

Tuttavia, potete utilizzare un clone Macintosh di Visual Basic chiamato (sorpresa!) Visual MacStandard Basic. Visual MacStandard Basic consente di disegnare interattivamente l'interfaccia utente e di scrivere codice BASIC affinché quest'ultima faccia qualcosa di utile.

Sebbene a Visual MacStandard Basic manchino molte delle funzioni avanzate di Visual Basic 6.0, tuttavia offre la possibilità di scrivere programmi per Macintosh in maniera rapida e semplice, ovvero proprio ciò che Visual Basic 1.0 ha fatto per Windows nel 1991. Quindi, se sapete programmare in Visual Basic, potrete utilizzare le vostre capacità di programmazione in Visual Basic per scrivere programmi Visual MacStandard Basic per Macintosh.

I più avventurosi possono sperimentare XBasic, un compilatore BASIC che viene eseguito sia sotto Windows, sia sotto Linux. Sebbene non sia intuitivo da utilizzare come Visual Basic, XBasic permette di scrivere programmi per Windows o Linux senza dover necessariamente imparare il C o il C++. Dato che Linux potrebbe diventare il sistema operativo di domani, potete sfruttare le vostre capacità di programmazione in BASIC scrivendo programmi Linux in modo da assicurarvi un impiego per il futuro.

Visual MacStandard Basic

ZCurve Software
8206 Rockville Road #280
Indianapolis, IN 46214
www.zcurve.com

XBasic

Basmark Corporation
P.O. Box 40450
Cleveland, OH 44140
Tel: 440-871-8855
Fax: 440-871-1715
www.basmark.com

Appendice A

A proposito del CD

Sempre che nessuno abbia preso una lametta e abbia tagliato la busta di plastica attaccata sulla copertina in fondo al libro, dovreste trovare un compact disc (CD o, più specificamente, un CD-ROM) racchiuso in una busta di plastica come una zanzara nell'ambra. Questo CD contiene il codice sorgente in Visual Basic tratto dai vari esercizi presentati nei capitoli del libro nonché alcune versioni dimostrative e programmi di prova extra forniti da terzi. Queste demo o programmi di prova sono programmi ActiveX dimostrativi che potete aggiungere ai vostri e che sono in grado di semplificare notevolmente la programmazione.

I programmi in Visual Basic mostrano esempi funzionanti dei comandi BASIC descritti nei vari capitoli. Studiando questo codice genuino in Visual Basic, potrete dedicare maggiore tempo all'apprendimento e meno tempo alla digitazione di noiosi comandi.

Requisiti di sistema

Assicuratevi che il vostro computer soddisfi i requisiti di sistema minimi presentati qui di seguito; in caso contrario, potreste avere difficoltà a utilizzare il CD.

- ✓ Un PC con un processore 486 o più veloce.
- ✓ Microsoft Windows 95/98/NT.
- ✓ Almeno 8MB di RAM installati sul computer. Per ottenere le prestazioni migliori possibili, io raccomando almeno 16MB di RAM (più RAM c'è, meglio è).
- ✓ Un drive CD-ROM - a doppia velocità (2x) o più veloce.
- ✓ Visual Basic versione 6.0 (le versioni precedenti non funzionano con il CD accluso).

Per ulteriori informazioni sui concetti di base, leggete *PC For Dummies*, quinta edizione, di Dan Gookin e *Windows 98 For Dummies* di Andy Rathbone (entrambi editi da IDG Books Worldwide, Inc.).

Come arrivare al sodo

Se state utilizzando Windows 95, 98 o NT, per arrivare ai programmi contenuti sul CD procedete come illustrato di seguito.

1. **Inserite il CD nel drive del CD-ROM del vostro computer.**

   Concedete al vostro computer un istante per dare uno sguardo al CD.

2. **Quando la lucina del drive del CD-ROM si spegne, fate doppio clic sul-l'icona Risorse del computer (che probabilmente si trova nell'angolo superiore sinistro del vostro desktop).**

   In questo modo verrà visualizzata la finestra Risorse del computer, che presenta tutti i drive collegati al computer, il pannello di controllo e qualche altra cosuccia utile.

3. Fate doppio clic sull'icona corrispondente al vostro drive del CD-ROM.

Per utilizzare il CD, procedete così.

1. **Fate doppio clic sul file chiamato License.txt.**

   Questo file contiene la licenza per l'utente finale che voi accettate nel momento in cui decidete di utilizzare il CD. Dopo aver letto la licenza, chiudete il programma (molto probabilmente NotePad) utilizzato per la visualizzazione del file.

2. **Fate doppio clic sul file chiamato Readme.txt.**

   Questo file contiene le istruzioni relative all'installazione del software a partire dal CD. Può essere opportuno lasciare questo file di testo aperto mentre utilizzate il CD.

3. **Fate doppio clic sulla cartella relativa al software che vi interessa.**

   Assicuratevi di leggere le descrizioni dei programmi contenute nel prossimo paragrafo della presente appendice (gran parte delle informazioni sono contenute anche nel file Readme). Queste descrizioni offrono informazioni più precise circa i nomi delle cartelle dei programmi nonché su come trovare ed eseguire l'installatore dei programmi.

4. **Trovate il file chiamato Setup.exe o Install.exe o qualcosa del genere e selezionatelo con un doppio clic.**

   L'installatore dei programmi vi accompagnerà nel processo di configurazione del vostro nuovo software.

Per eseguire alcuni dei programmi inclusi sul CD-ROM *Visual Basic 6 For Dummies*, potrebbe essere necessario tenere il CD nel drive del CD-ROM e Questa è una Buona Cosa. In caso contrario, il programma installato dovrebbe chiedervi di

installare su disco fisso un'ampia porzione del programma, impedendovi così di installare altro software.

Cosa trovate

Il codice di esempio in Visual Basic è contenuto in cartelle separate, definite in base al capitolo di appartenenza (per scoprire cosa c'è in ogni cartella, date uno sguardo alla Tabella A.1). Per esempio, per trovare tutto il sorgente in Visual Basic relativo al Capitolo 3, cercate sul CD una cartella chiamata Chapter 3. Potete caricare ed eseguire il codice sorgente direttamente dal CD oppure copiarlo su disco fisso (ricordate solo che se decidete di eseguire il codice sorgente dal CD, non potrete modificarlo).

Tabella A.1 Contenuto delle cartelle

Cartella	*Che cosa contiene*
Capitolo 3	Contiene due programmi, Hello1 e Hello2. Caricando questi file dal disco, non sarete costretti a copiare tutto il codice dal libro.
Capitolo 4	Contiene il programma Hello finale che potrete esaminare e apprezzare.
Capitolo 5	Contiene il programma Caption che dimostra in che modo il codice BASIC può modificare il titolo di un oggetto visualizzato sull'interfaccia utente.
Capitolo 6	Contiene il programma Forms and Button che mostra come utilizzare i pulsanti di comando per modificare l'aspetto di una form (come extra, il programma mostra anche come utilizzare le finestre di dialogo, un argomento trattato nel Capitolo 13).
Capitolo 7	Contiene il programma Boxes and Buttons che mostra come utilizzare le caselle di controllo, i pulsanti di opzione, le caselle combinate e le caselle di riepilogo.
Capitolo 8	Contiene il programma TextBoxes che mostra come utilizzare una casella di testo destinata alla parola chiave e una casella di testo normale.
Capitolo 9	Contiene il programma ScrollBars che mostra come utilizzare le barre di scorrimento e visualizzare sullo schermo il valore della barra.

(continua)

Tabella A.1

Cartella	*Che cosa contiene*
Capitolo 10	Contiene il programma Shapes che mostra come modificare un cerchio durante il periodo di esecuzione a partire dall'interfaccia utente.
Capitolo 13	Contiene il programma DialogBox che mostra come utilizzare l'oggetto incorporato Visual Basic Common Dialog Box.
Capitolo 15	Contiene il programma Variables che mostra come modificare le proprietà degli oggetti memorizzati su varie form.
Capitolo 16	Contiene i programmi ListBoxes che mostrano come utilizzare una casella di riepilogo Visual Basic.
Capitolo 17	Contiene il programma Precedence che mostra come funziona la precedenza quando vengono utilizzate e quando invece vengono omesse le parentesi durante il calcolo di un risultato numerico.
Capitolo 18	Contiene il programma ELIZA che mostra come manipolare le stringhe. ELIZA mima uno psicoterapeuta che si limita a ripetere tutto ciò che digitate, dandovi l'illusione che il computer vi stia rispondendo.
Capitolo 23	Contiene il programma DoWhile che mostra come funziona un semplice ciclo Do While.
Capitolo 25	Contiene il programma ForNext che mostra come funziona un semplice ciclo ForNext.
Capitolo 28	Contiene il programma Arguments che mostra come passare gli argomenti da una procedura all'altra.
Capitolo 29	Contiene il programma Functions che mostra come opera una funzione che converte le iarde in metri.
Capitolo 30	Contiene il programma Mission, che mostra come funzionano i moduli di classe e come utilizzarli nell'ambito di un programma in Visual Basic.
Capitolo 34	Contiene il programma PrintExample che mostra come stampare dall'interno di un programma in Visual Basic.
BonusGame	La cartella BonusGame contiene il codice sorgente in Visual Basic per una versione del famoso gioco Pong. Potete utilizzare liberamente questo codice sorgente nei vostri programmi oppure potete dissezionare ed esaminare il programma per capire come creare effetti di animazione in Visual Basic.

Che cosa c'è nella cartella ActiveX

La cartella ActiveX contiene tutti i programmi di prova ActiveX (per tutti i dettagli, leggete la Tabella A.2) gentilmente offerti da società terze desiderose di mostrarvi i loro prodotti e ansiose di vendervi le versioni complete dei loro programmi. Sebbene siate liberi di fare esperimenti su queste versioni, non potete utilizzarle nei vostri programmi in Visual Basic che intendete vendere.

Ogni programma di prova ActiveX comprende il proprio programma di setup (di solito definito SETUP.EXE). Per esempio, per eseguire questo programma è sufficiente che facciate doppio clic sul file SETUP.EXE in Windows Explorer oppure che facciate clic sul pulsante Avvio sulla barra di Windows, che selezioniate Esegui e quindi Browse finché non avrete trovato il programma SETUP.EXE che intendete eseguire.

Alcuni dei programmi inclusi sul CD Visual Basic 6 For Dummies sono stati sviluppati prima del rilascio di Visual Basic 6. Questi programmi sono contrassegnati come in grado di funzionare solo con Visual Basic 4 e/o Visual Basic 5. In ogni caso, tutto il software incluso sul CD è stato collaudato e si installa e può essere

Tabella A.2 Contenuto della cartella ActiveX

Programma	*Cosa fa*
Crescent Internet ToolPak 4.1	Se intendete creare applicazioni Internet, provate questa versione dimostrativa di Crescent Internet ToolPak, che aiuta gli sviluppatori VB a creare applicazioni utilizzabili per il browsing del World Wide Web, inviare posta elettronica, eseguire trasferimenti di file FTP e molto altro ancora. Il Crescent Internet ToolPak semplifica l'accesso al brodo primordiale alfabetico degli standard Internet come IMAP4, TCP/IP Server e FTP proxy.
Elastic Light 6.0	Questa versione dimostrativa può fare in modo che i vostri programmi si adattino automaticamente alla risoluzione dello schermo. Quindi, se avete scritto il vostro programma su un monitor super VGA con una risoluzione di 800 x 600 dpi e poi qualcuno lo esegue su un vecchio monitor VGA con una risoluzione di 640 x 480 dpi, il programma non uscirà dallo schermo. Questo controllo ActiveX regola anche le dimensioni degli oggetti sulla form (come per esempio i pulsanti di comando e i pulsanti di opzione) qualora l'utente dovesse decidere di ridimensionare la form.

(continua)

Tabella A.2

Programma	*Cosa fa*
HASHcipher	Nel caso aveste bisogno di segretezza, provate questa versione dimostrativa di HASHcipher. Questo controllo ActiveX utilizza il Secure Hash Standard (SHS), sviluppato dall'Ente per la Sicurezza Nazionale del Governo USA (NSA), per garantire la sicurezza mediante l'autenticazione. L'algoritmo specificato dallo standard, il Secure Hash Algorithm (SHA-1), è considerato da molti crittografi come l'algoritmo più resistente disponibile oggi e può essere utilizzato in qualsiasi applicazione che richieda l'autentificazione di un file o di un messaggio.
InstallShield Express 2.02	Offre un programma di installazione più sofisticato rispetto a quello fornito con Visual Basic. InstallShiled è il programma di installazione più diffuso, utilizzato dagli sviluppatori commerciali di ogni luogo, quindi unitevi alla banda e sperimentate subito questa versione di prova di InstallShield!
True DBList	La versione di prova di True DBList offre una serie di caselle combinate e di riepilogo personalizzabili per accedere alle informazioni contenute nei file di database. Supporta più colonne, intestazioni di gruppo divise, più righe per record, colori o stili alternati per le righe, immagini grafiche e testo inseriti nelle celle, colori e font dipendenti dai dati nonché migliaia di altre funzioni che potrebbero sembrarvi assolutamente prive di significato finché non avrete provato l'atroce esperienza di accedere ai dati con le funzioni incorporate di Visual Basic.
True DBInput	La versione di prova di True DBInput offre controlli di input ActiveX consapevoli dei dati per la creazione di applicazioni database in Visual Basic. True DBInput comprende decine di funzioni per gestire l'immissione e la validazione dei dati, la visualizzazione personalizzata e intuitiva di data, ora, testo e numeri nonché altre funzioni utili per assicurarvi che i vostri utenti non digitino informazioni errate nei vostri file di database.
VBAssist 5.0	La versione di prova di VBAssist contiene vari strumenti per aiutarvi a scrivere programmi in Visual Basic più velocemente e facilmente di prima. Con VBAssist potrete attribuire con rapidità Tab Order ai controlli su una form, tasti rapidi o acceleratori ai controlli, sistemare i controlli nelle posizioni corrette di una form oltre a numerose altre funzioni utili per risparmiare tempo.

Tabella A.2

Programma	*Cosa fa*
VBPartner	Questa edizione di prova di VBPartner è un add-in integrato di Microsoft Visual Basic che include 11 "partner" unici utili per risparmiare tempo che vi aiuteranno con le attività di routine come la progettazione delle applicazioni, la gestione degli errori e molto altro ancora semplificando noiosi compiti di sviluppo.
VSData	Questa versione dimostrativa di VSData consente alle vostre applicazioni Visual Basic di accedere ai file di database senza utilizzare gli ingombranti file di database propri di Visual Basic che fagocitano spazio su disco e offrono molte più funzioni di quelle di cui necessita il vostro programma.
VSDirect	Questa versione dimostrativa di VSDirect fornisce delle routine per accedere alla tecnologia DirectX di Microsoft per la creazione di immagini grafiche, suoni e animazione. Studiato prevalentemente per la creazione di applicazioni multimediali, VSDirect include anche ruotine per eseguire giochi servendosi di un collegamento modem o di una rete.
VSDocX	Questa versione dimostrativa di VSDocX aiuta a creare file di guida e documentazione stampata per le vostre applicazioni Visual Basic. Vi libera dal fardello di dover scrivere voi la documentazione del programma, un compito odiato dalla maggior parte dei programmatori, come dimostra la scarsa qualità dei manuali di informatica di tutto il mondo.
VSFlex 3.0	Visual Basic viene fornito con un controllo ActiveX chiamato MXFlexGrid, una versione più vecchia di VSFlex. Analogamente a VSFlexGrid, questa versione dimostrativa di VS-Flex consente di ordinare, fondere e raggruppare i dati in una griglia. VSFlex inoltre include un modello di stringa che corrisponde al controllo ActiveX per la valutazione delle espressione matematiche o l'aggiunta ai vostri programmi di capacità linguistiche.
VS-OCX 6.0	Offre schede stile notebook unitamente a un controllo ActiveX per il parsing delle stringhe per una manipolazione delle stringhe a livello avanzato. Questa versione dimostrativa di VS-OCX include anche un sofisticato controllo di adattamento alla risoluzione dello schermo (simile ad ElasticLight).

(continua)

Tabella A.2

Programma	*Cosa fa*
VSReports	VSReports consente ai vostri programmi in Visual Basic di creare e stampare report dai file di database in Microsoft Access. Utilizzando questa versione dimostrativa di VSReports, potrete evitare di utilizzare gli ingombranti e voluminosi file di reporting per i database propri di Visual Basic.
VSSpell	Offre un controllo dell'ortografia e un dizionario dei sinonimi per le vostre applicazioni in Visual Basic. È perfetto per creare la prossima generazione di programmi di elaborazione testi che soppianteranno la Microsoft.

eseguito con Visual Basic 6. Visitate il sito Web del produttore di ciascun programma (come indicato sul file Readme presente sul CD) per gli aggiornamenti a Visual Basic 6.

In caso di problemi (riguardanti il CD)

Dato che i computer sono famosi per essere pedanti, pignoli e assolutamente inaffidabili nei momenti cruciali della nostra vita, potreste rientrare tra i pochi sfortunati che andranno incontro a problemi quando tenteranno di utilizzare il CD accluso.

Se il vostro computer non riesce a leggere i file contenuti sul CD, è possibile che il CD sia difettoso. Nonostante i nostri migliori sforzi, ogni tanto un CD decide di seguire "il lato oscuro della Forza", di dare messaggi di errore e di spaventare la gente facendole credere che sia il computer il responsabile di tutto.

Se il vostro computer legge gli altri CD ma non quello fornito insieme al libro, contattate il Servizio Assistenza Clienti di IDG Books Worldwide, Inc. al numero 800-762-2974 e chiedete un CD sostitutivo.

Se non avete Visual Basic 6.0, non potete caricare i programmi campione contenuti sul CD e questo anche se possedete versioni precedenti di Visual Basic, come la versione 5.0 o 4.0. Quindi, controllate la vostra versione di Visual Basic selezionando Help-About Visual Basic. In questo modo verrà visualizzata una finestra di dialogo dove troverete quale versione di Visual Basic avete.

Indice analitico

• C •

• D •

• E •

• p •

• T •

• U •

• V •

W

Dal Catalogo Apogeo – PROGRAMMAZIONE